IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Masayuki Masuda, et al.

Attorney Docket No.: OMRNP018

Application No.: 10/091,113

Examiner: To Be Assigned

Filed: February 28, 2002

Group: 2175

Title: CONTROLLERS, TOOLS AND SYSTEMS COMPRISING SAME

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail to: Commissioner for Patents, Washington, DC 20231 on December 10, 2003.

Signed: Deborah Neill

Deborah Neill

TRANSMITTAL OF CERTIFIED COPY OF PRIORITY DOCUMENT

Commissioner for Patents
Washington, D.C. 20231

Sir:

Enclosed herewith are certified copies of priority documents Japan patent application No. 2001-068288 filed on March 12, 2001. Please file this document in the subject application.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

Kenichi Nishimura
Registration No. 29,093

P.O. Box 778
Berkeley, CA 94704-0778
(510) 843-6200

日　本　国　特　許　庁
JAPAN　PATENT　OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application:　２００１年　３月１２日

出願番号
Application Number:　特願２００１－０６８２８８

[ST.10/C]:　[ＪＰ２００１－０６８２８８]

出願人
Applicant(s):　オムロン株式会社

２００２年　２月１９日

特許庁長官
Commissioner,
Japan Patent Office　及川耕造

出証番号　出証特２００２－３００７９５２

【書類名】　特許願

【整理番号】　59756

【提出日】　平成13年 3月12日

【あて先】　特許庁長官殿

【国際特許分類】　G05B 19/05

【発明者】

【住所又は居所】　京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地　オムロン株式会社内

【氏名】　益田　真之

【発明者】

【住所又は居所】　京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地　オムロン株式会社内

【氏名】　長尾　嘉祐

【発明者】

【住所又は居所】　京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地　オムロン株式会社内

【氏名】　門脇　正規

【発明者】

【住所又は居所】　京都府京都市下京区塩小路通堀川東入南不動堂町８０１番地　オムロン株式会社内

【氏名】　工藤　敏巳

【特許出願人】

【識別番号】　000002945

【氏名又は名称】　オムロン株式会社

【代表者】　立石　義雄

【代理人】

【識別番号】　100092598

【弁理士】

【氏名又は名称】　松井　伸一

【手数料の表示】

【予納台帳番号】　019068

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9800459

【プルーフの要否】　要

【書類名】　　明細書

【発明の名称】　コントローラ及びツール並びにそれらにより構成されるシステム

【特許請求の範囲】

【請求項１】　コントローラに接続されるデバイスを管理するための制御プログラムと、

前記制御プログラムと前記デバイスを関連付けた関連情報を記憶する関連情報記憶手段とを備え、

前記制御プログラムが前記デバイスにアクセスするに際し、前記関連情報記憶手段に格納された関連情報を参照することによりアクセス先を特定し、実行するようにしたことを特徴とするコントローラ。

【請求項２】　接続されたデバイスと通信を行い、前記アクセス先を決定するとともに、前記関連情報記憶手段に格納する通信処理手段を備えたことを特徴とする請求項１に記載のコントローラ。

【請求項３】　前記デバイスと前記コントローラのデータの送受は、前記コントローラ内に設置されたコントローラメモリの所定領域を介して行うものであり、

前記デバイスのメモリサイズに基づき、前記コントローラメモリに対するメモリ割付を行うとともに、その割付結果を前記関連情報として登録する割付処理手段を備えたことを特徴とする請求項１または２に記載のコントローラ。

【請求項４】　前記デバイスは、動作するために必要な動作情報を保持するものであり、

動作終了の際に、前記動作情報を取得するとともに、記憶保持し、動作開始の際に、前記記憶保持した動作情報を前記デバイスに対してダウンロードする機能を備えたことを特徴とする請求項１～３のいずれか１項に記載のコントローラ。

【請求項５】　前記デバイスと前記コントローラのデータの送受は、前記コントローラ内に設置されたコントローラメモリの所定領域を介して行うものであり、

前記コントローラメモリに前記デバイスの異常情報を格納する領域を設定し、

前記異常情報と、前記関連情報に基づき、異常のあったデバイス用の保守情報を出力することを特徴とする請求項１～４のいずれか１項に記載のコントローラ。

【請求項６】　請求項１～５に記載のコントローラに接続可能なツールであって、

作成した制御プログラムと、その制御プログラムが管理するデバイスの情報から、基本関連情報を作成する機能と、

その作成する機能で作成された前記基本関連情報を前記コントローラに向けてダウンロードする機能を備えたことを特徴とするツール。

【請求項７】　請求項１～５に記載のコントローラと、請求項６に記載のツールを備えたシステムであって、

前記ツールで作成された基本関連情報を前記コントローラにダウンロードし、

前記コントローラは、取得した基本関連情報に基づいて、前記関連情報を生成し、前記記憶手段に格納するようにしたことを特徴とするシステム。

【請求項８】　制御プログラムと、その制御プログラムによって管理されるデバイスを関連付けた関連情報を記憶する関連情報記憶手段を備えたコントローラにおける処理方法であって、

前記制御プログラムの実行にともない前記デバイスにアクセスする場合、前記関連情報記憶手段に格納された関連情報を参照して前記デバイスのアクセス先を特定する処理を実行後、

その特定したアクセス先のデバイスにアクセスすることを特徴とするデバイスへのアクセス方法。

【請求項９】　コントローラに接続されるデバイスを管理するための制御プログラムと、

前記制御プログラムと前記デバイスを関連付けた関連情報とを対にし、

あるコントローラで利用している制御プログラムを別のコントローラで再利用する場合には、その制御プログラムとともに、対となった前記関連情報も前記別のコントローラに実装し、

次いで、前記関連情報中のデバイスのアクセス先に関する情報を前記別のコントローラにおけるデバイスのアクセス先に合わせて修正するようにした制御プログラムの再利用方法。

【請求項１０】　コントローラに接続されるデバイスを管理するための制御プログラムと、

前記制御プログラムと前記デバイスを関連付けた関連情報を記憶する関連情報記憶手段とを備え、

前記制御プログラムが前記デバイスにアクセスするに際し、前記関連情報記憶手段に格納された関連情報を参照することによりアクセス先を特定し、実行するコントローラにおける関連情報の管理方法であって、

前記関連情報に、前記デバイスの設定パラメータを含む場合、

前記コントローラの起動時に前記関連情報に記憶された前記設定パラメータを前記デバイスにダウンロードし、

前記コントローラの終了時に前記デバイスに設定されている設定パラメータをアップロードして記憶保持するようにしたことを特徴とする関連情報の管理方法。

【請求項１１】　コントローラに接続されるデバイスを管理するための制御プログラムと、

前記制御プログラムと前記デバイスを関連付けた関連情報を記憶する関連情報記憶手段とを備え、

前記制御プログラムが前記デバイスにアクセスするに際し、前記関連情報記憶手段に格納された関連情報を参照することによりアクセス先を特定し、実行するコントローラ内のコントローラメモリ上に、デバイスのステータス情報を記憶する領域をデバイスごとに設定し、

前記コントローラは、前記領域を監視し、異常を検知した場合に、前記関連情報記憶手段をアクセスして異常を発生しているデバイスの関連情報を取得し、所定の情報を外部周辺機器に通知することを特徴とする異常監視方法。

【請求項１２】　請求項１から５のいずれか１項に記載のコントローラと、デバイスが接続されたネットワークに情報処理装置を接続し、

前記情報処理装置は、前記コントローラに実装された前記制御プログラム及び関連情報と同等のものを備えるとともに、前記コントローラと前記デバイス間の通信状態を参照する機能を備え、

前記参照した通信状態に従い、前記情報処理装置内の制御プログラムを実行して、前記コントローラと前記デバイスからなるシステム動作状況を監視等の情報処理を行い、

その結果を前記コントローラに通知する情報処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明は、コントローラ及びツール並びにそれらにより構成されるシステムに関するものである。

【０００２】

【発明の背景】

良く知られているように、ＦＡなどの工業領域におけるコントロールシステムには、ＰＬＣなどの制御装置（コントローラ）と、その制御装置により制御される機器（デバイス）を適宜組み合わせて構築される。図１に示すように、コントローラ１には、ネットワーク２を介して１または複数のデバイス３ａ，３ｂが接続されている。またコントローラ１の内部には、各デバイス３ａ，３ｂをハンドリングするための制御プログラム（ドライバ）４ａ，４ｂが実装されている。この制御プログラム４ａ，４ｂは、デバイス３ａ，３ｂごとに用意され、Ｉ／Ｏメモリ５を利用しながら対応するデバイス３ａ，３ｂとデータの送受を行い、所定の処理を実行する。また、図示の例では、制御プログラム（Ｘ）６は、制御プログラム４ａ，４ｂを呼び出すもので、全体として１つのタスクが構成されている。

【０００３】

なお、デバイス３ａ，３ｂは、例えば、ＯＮ／ＯＦＦセンサやデジタルＩ／Ｏのように、ＯＮ／ＯＦＦ情報をコントローラ１（ドライバ：制御プログラム４ａ，４ｂ）との間で通信するものもあれば、光電センサ（パラメータは光量，閾値

），温度調節器（パラメータは設定温度，ＰＩＤ値など）などのように、デバイス内で設定すべきパラメータを持ち、ユーザが作成したプログラムが動作するものがある。

【０００４】

また、図１に示した例では、各デバイス３ａ，３ｂはネットワーク２を介してコントローラ１に接続されているが、デバイスとコントローラ間の通信を、コントローラ１内のバスを介して行われる場合もある。例えばコントローラ１としてビルディングブロック式のＰＬＣを用い、そのベースユニット上のスロットにモーションコントロールユニットやＰＩＤ制御ユニットを直接接続した場合である。

【０００５】

一方、上記した通信を行うためには、制御プログラム４ａ，４ｂと、その制御プログラム４ａ，４ｂがアクセスすべきデバイス３ａ，３ｂを関連付ける必要があり、通常、デバイス３ａ，３ｂに対してネットワーク２上にユニークに設定される識別子としてのノード番号を付与し、そのノード番号を指定してアクセスする。なお、デバイスがバス接続されている場合には、ノード番号はユニット番号となるが、基本的な考えは同じである。

【０００６】

図１に示す例では、デバイス（Ａ）３ａのノード番号は＃５となり、デバイス（Ｂ）３ｂのノード番号は＃６となっている。従って、制御プログラム（Ａ）４ａが、デバイス（Ａ）３ａに対してメッセージ通信でアクセスする場合、プログラム中にデバイス（Ａ）３ａのノード番号（＃５）を記述することになる。

【０００７】

ところで、一度作成した制御プログラムを再利用することができると、新たに制御プログラムを作成しなくて済むことがあり、開発に係る労力が軽減される。一方、ドライバと称される制御プログラムは、その制御プログラム（ソフト）がアクセスするデバイス（ハード）と一体となって初めて機能を発揮するものであり、両者を一体として取り扱った方が好ましい。なお、以下の説明では、それら制御プログラムとデバイスを一体にしたものを制御オブジェクトと称する。

【０００８】

しかしながら、従来のシステムでは、プログラム単位での再利用はできたものの、制御プログラムとデバイスを一体として扱う方法がなく、それぞれを別々に再利用することになり、再利用が煩雑である。

【０００９】

また、図１に示すように、例えば、生産ラインの変更や、補強，修理のために、一方のコントローラ１で使用していたデバイス（Ａ）３ａを別のコントローラ１′に接続して使用したいという要求がある。このような場合に、上記した制御オブジェクトの考えに従い、制御プログラムとデバイスをともにコントローラ１′に移行（再利用）することを考える。すると、まず、制御プログラムは、プログラミングツールやコントローラに装備されているメモリカードなどを経由して、容易にコントローラにダウンロードすることができる。また、デバイス（Ａ）３ａをハード的にコントローラ１′が接続されているネットワーク２′に連結することにより移行できる。

【００１０】

しかし、そのように単純に接続しただけでは、制御オブジェクトを動作させることはできない。すなわち、制御プログラムと対になるデバイスとの間でコミュニケーションを行うためのコンフィグレーションが必要であるが、元のコントローラ１で使用していた情報をそのまま使用すると動作しなくなってしまう。

【００１１】

一例を示すと、図１に示したものの場合、制御オブジェクト（Ａ）４ａが管理するデバイス（Ａ）のノード番号は，＃５であったため、制御プログラム（Ａ）の中、或いは制御プログラム（Ａ）を呼出す制御プログラム（Ｘ）の中に、デバイス（Ａ）にアクセスするためにノード番号（＃５）を指定するプログラム部分がある。

【００１２】

一方、コントローラ１′では、ノード番号（＃５）は既に他のデバイスに使用されている。従って、制御プログラム（Ａ），（Ｘ）において、ノード番号（＃５）とある部分をそのままにしていると、エラーとなる。従って、図示の場合に

は、該当する部分のプログラムを正しいノード番号である（＃２）に書き替える必要があるが、該当する部分を全てピックアップし、修正すること、並びに係る修正が他のプログラムへ影響を与えないことを注意，確認しながら実行することは煩雑である。

【００１３】

さらに、デバイス（Ａ）のコントローラ１，１′のＩ／Ｏメモリへの割付（Ｉ／Ｏコンフィグレーション）も、それぞれ異なることがある。そして、制御プログラム（Ａ）におけるデバイス（Ａ）の割付けられているＩ／Ｏメモリ５へのアクセス方法が、アドレスを指定するようなプログラミングスタイルの場合や、たとえ変数プログラミングであっても、上記Ｉ／Ｏメモリ５に配列でアクセスするような場合には、再利用の際に係る割り付けが変更されたことを考慮し、該当するプログラム部分を変更する必要があるので、やはり煩雑である。そして、当然のことながら、アドレスを間違えると、動作ができなくなる。

【００１４】

さらにまた、上記したように、制御オブジェクトを再利用するに際し、Ｉ／Ｏメモリ５への再割り付けが必要な場合、さらに以下に示す問題がある。すなわち、開発者は、再利用先のコントローラ１′のＩ／Ｏメモリ５の使用状況、つまり、「どこに、どれだけの空きエリアがあるか」などのメモリマップを把握している必要があり、空きエリアを効率良く使用するように割り付けを行い、実際にプロクラムを組む必要がある。そのため、割付ミス等人為的なミスを起こしやすい。

【００１５】

さらに、必要に応じて、既に接続されたデバイス用に割り付けたＩ／Ｏ割付けまで変更されることがあり、係る場合には、そのすでに接続されているデバイス用の制御プログラムまで変更しなければならず、煩雑であるばかりでなく、割り付けミスをする可能性が高くなる。

【００１６】

さらにまた、デバイスの中には、光電センサ，温度調節器などのように、デバイス内で設定すべきパラメータを持つものがあるのは、既に述べたが、係るパラ

メータやプログラムをデバイスに設定するためには、ツール装置をデバイスに直接接続したり、或いは、コントローラに接続し、そのコントローラを介して各種の設定を行う必要がある。

【００１７】

従って、仮にデバイスが故障した場合、そのデバイスを交換することになるが、新たなデバイスに対して、再度故障前のデバイスに対して行ったパラメータ等の設定処理と同じ工程を実施することになり、非常に工数がかかる。係る処理は、マニュアルで調整・設定することになるが、前回行った設定情報をどこかの記憶媒体や帳票に記憶しておき、その設定情報を見ながら操作員が設定することになる。従って、入力処理時の人為的ミスが発生しやすい。また、設定情報を記憶した帳票等が見つからなかったり、違う情報を見て設定してしまうおそれなどもある。

【００１８】

一方、図２に示すようなシステム構成では、デバイス３で異常が発生した場合、デバイス３に割り付けられているコントローラ１のＩ／Ｏメモリ５の所定ビットが変化する。すると、そのコントローラ１のＩ／Ｏメモリ５を参照している表示器７が、そのビットの変化を検出し、異常ありと認識することにより異常画面を表示するようになっている。係る異常画面を表示するための具体的な処理は、以下の通りである。

【００１９】

すなわち、コントローラ１のシステムソフト（第１通信処理８ａ）がデバイス３の状態を検出し、コントローラ１のＩ／Ｏメモリ５上の予め定めた所定アドレスに格納する。また、コントローラ１内の制御プログラム４は、上記したＩ／Ｏメモリ５の所定アドレスを監視し、ビットが立つか否かを判断する。そして、ビットが立ったことを確認すると、制御プログラム４は、Ｉ／Ｏメモリ５の別のアドレスにビットを立てる。このビットが立ったならば、制御プログラム４は、デバイスの状態をもとに、「表示器に表示させたい画面番号／数値データ（異常コード）／文字列データ（異常内容）」等の情報をメモリ５上の所定領域に格納する。係る情報は、第２通信処理８ｂを介して表示器７に送られる。表示器７側で

は、通信処理部８ｃが受け取った情報に基づき表示器７内のメモリ５に格納する。画像表示処理部４ｃでは、そのメモリ５に格納された情報に伴い、表示器７で表示すべき画面番号を獲得し、画面メモリ９から対応する画面を呼び出すとともに、その呼び出した画面を表示する。

【００２０】

ところで、上記の表示される内容は、あるデバイスで異常が発生しても、そのデバイスがシステムにおいてどのような意味を持っているものかまではわからない。例えば、ネジ締めロボットという論理単位をユーザは認識しているが、その構成要素のデバイスの通信アドレスが異常、或いは、デバイスが割り付いているコントローラのあるビットのＯＮ／ＯＦＦで故障と通知しても、どういう制御要素で異常が発生したか理解することが容易でない。そのため、何らかの表をもとに論理的な意味を理解し、保守，メンテナンスを行うことになるので、その理解に時間がかかり、その結果、保守，運用に時間を要する。

【００２１】

更にまた、生産ラインは、複数の設備から構成されており、各設備には、表示器，パネルコンピュータなど、いわゆるヒューマン・マシン・インターフェース（以下、ＨＭＩ）が装備され、その設備の稼動状況や設備を構成するデバイスの情報，生産情報，設備の操作のために使われる。また、生産ライン／工場自体の生産情報などをモニタするためのパソコン（これもＨＭＩと言える）が工場内、或いは工場外に設置されている。

【００２２】

ところで、係るＨＭＩに表示される数値，文字，絵といった情報は以下のように決められる。すなわち、コントローラに接続されているデバイスの値を表示する際には、デバイスのメモリがコントローラのメモリに割り付いている場合、デバイスが割り付いているコントローラのメモリをＨＭＩが通信によって参照し、その値をＨＭＩで表示する。もし、上記値に演算が必要な場合には、一般に情報系のプログラムを実行するのに適したＨＭＩ内で演算処理を行う。

【００２３】

しかしながら、ＨＭＩのプログラム・表示画面を設計する開発者、或いはその

ＨＭＩを見る保守員は、設備を構成するデバイスについては、理解容易性のためにオブジェクト名でアクセスして情報を収集したいという要求がある。しかし、従来のシステムでは、ユーザがデバイスをＩ／Ｏコンフィグレーションでコントローラにデバイスのメモリをマッピングし、その情報を把握した上で、ＨＭＩの設定ツールで、コントローラのメモリのアドレスを指定する必要がある（図２参照）。

【００２４】

従って、設備変更等が発生して上記のデバイスのメモリマップが変更された場合には、ＨＭＩの演算プログラムも書き替える必要がある。つまり、仮にコントローラ（制御系）のプログラム開発と、情報系のプログラムを平行して行うような場合には、常に相手側の開発状況に注意をはらい、必要に応じて相手方の変更に伴う自己の開発プログラムの変更を行わなければならず、開発処理が煩雑で長期化する。

【００２５】

この発明は、制御プログラムとデバイスを一体でオブジェクト（制御オブジェクト）として扱うことができ、再利用が容易かつ確実に行え、再利用時に関係の無い制御オブジェクトへ影響を与えることが無く、また、デバイスの通信アドレスや、メモリ割付が変更された場合に、その通信アドレス，メモリ割付等が変更のデバイスにアクセスする制御プログラムや、情報系のプログラムに影響を与えることがなく、情報系のプログラムの作成と制御プログラムの作成の独立性を確保でき、さらに、デバイスの交換の際に発生するパラメータの設定の迅速化，正確化を図り、また、デバイス，コントローラで発生した異常をユーザフレンドリに通知することのできるコントローラ及びツール並びにそれらにより構成されるシステムを提供することを目的とする。

【００２６】

【課題を解決するための手段】

この発明によるコントローラでは、コントローラに接続されるデバイスを管理するための制御プログラムと、前記制御プログラムと前記デバイスを関連付けた関連情報を記憶する関連情報記憶手段とを備え、前記制御プログラムが前記デバ

イスにアクセスするに際し、前記記憶手段に格納された関連情報を参照することによりアクセス先を特定し、実行するように構成した。

【００２７】

また、本発明のアクセス方法では、制御プログラムと、その制御プログラムによって管理されるデバイスを関連付けた関連情報を記憶する関連情報記憶手段を備えたコントローラにおける処理方法の一つであって、前記制御プログラムの実行にともない前記デバイスにアクセスする場合、前記関連情報記憶手段に格納された関連情報を参照して前記デバイスのアクセス先を特定する処理を実行後、その特定したアクセス先のデバイスにアクセスするようにしている。

【００２８】

「デバイスを管理する」とは、デバイスと関連付けられ、デバイスを制御、つまり、デバイスが入力デバイスの場合には、その入力デバイスからの情報を取得して所定の処理を実行し、また、デバイスが出力デバイスの場合には、そのデバイスに対して命令を送り所定の動作を行わせることを意味する。広い意味では、デバイスと情報の通信を行う制御プログラムは、デバイスを管理するための制御プログラムとなる。

【００２９】

関連情報は、少なくとも制御プログラムがデバイスに対してアクセス可能とするための情報であり、制御プログラム側でデバイスを特定するための情報（デバイス名など）と、実際にアクセス先を特定する通信アドレス（ノード番号）であったり、コントローラメモリに割付けられたメモリアドレス等を含む。このようにすることにより、制御プログラム側では、デバイス名などのデバイスを特定するための情報を用いてプログラムを組み、デバイスを呼び出す際には、関連情報記憶手段に格納されたそのデバイスについての関連情報を参照することにより、アクセスができる。

【００３０】

よって、再利用などによりデバイスの通信アドレスや、割り付けられたメモリアドレスが変換されたとした場合には、関連情報を修正するだけで対応できる。つまり、制御プログラムを修正する必要がない。その結果、デバイスと制御プロ

グラムを一体として扱い、移動なども簡単に行える。

【００３１】

また、ＨＩＭなどの情報系の機器がデバイスにアクセスしたり、動作状況を監視する場合にも、上記の関連情報を参照することにより、対応できる。よって、仮に制御系のプログラム変更などがあったり、デバイスのアクセス先が変わったとしても、情報系のプログラムに影響はない。従って、制御系と情報系のプログラムを平行して開発ができる。

【００３２】

関連情報には、具体的なアクセス先（アドレス）などを格納することになるが、係る情報をマニュアルにより開発者等が入力しても良いが、例えば、接続されたデバイスと通信を行い、前記アクセス先を決定するとともに、前記関連情報記憶手段に格納する通信処理手段を備えたり、前記デバイスと前記コントローラのデータの送受は、前記コントローラ内に設置されたコントローラメモリの所定領域を介して行うものであり、前記デバイスのメモリサイズに基づき、前記コントローラメモリに対するメモリ割付を行うとともに、その割付結果を前記関連情報として登録する割付処理手段を備えるようにすると、自動的に設定できるので、簡単かつ正確に行える。

【００３３】

また、デバイスの中には、動作するために必要な動作情報（実施形態では、「設定パラメータ」）を保持し、それに基づいて所定の動作をするものがある。係るデバイスに対しては、動作終了の際に、前記動作情報を取得するとともに、記憶保持し、動作開始の際に、前記記憶保持した動作情報を前記デバイスに対してダウンロードする機能を備えるとよい。これにより、稼動中に動作情報が変更された場合であっても、終了時に係る動作情報をコントローラ側に記憶することにより、実際のデバイスの動作情報と、コントローラ側での記憶内容が整合された状態が保証され、動作情報の管理が容易に行える。

【００３４】

前記デバイスと前記コントローラのデータの送受は、前記コントローラ内に設置されたコントローラメモリの所定領域を介して行うものであり、前記コントロ

ーラメモリに前記デバイスの異常情報を格納する領域を設定し、前記異常情報と、前記関連情報に基づき、異常のあったデバイス用の保守情報を出力するようにするとよい。異常のあったデバイスを知らせるだけでなく、そのデバイスの保守をする際の補助情報を出力することにより、効率的に保守作業が行える。ここで、保守情報は、実施の形態では、「異常が発生した制御オブジェクト名，異常が発生したデバイス名，異常が発生したデバイスの通信アドレス並びに異常原因等」である。これにより、故障したデバイスがどの制御オブジェクトに該当するか等の情報を知ることができ、迅速に保守作業が行える。

【００３５】

そして、本発明に係るツールは、上記した各コントローラに接続可能なツールであって、作成した制御プログラムと、その制御プログラムが管理するデバイスの情報から、基本関連情報を作成する機能と、その作成する機能で作成された前記基本関連情報を前記コントローラに向けてダウンロードする機能を備えている。

【００３６】

さらに、本発明に係るシステムは、上記した各コントローラと、ツールを備えたシステムである。そして、そのツールで作成された基本関連情報を前記コントローラにダウンロードし、前記コントローラは、取得した基本関連情報に基づいて、前記関連情報を生成し、前記記憶手段に格納するようにした。

【００３７】

実施の形態では、基本関連情報は、具体的なデバイスを特定するシリアルナンバーや、アドレス（通信，メモリ）が空欄で、そのデバイスの一般的な情報と制御プログラムの関係を格納するようにし、コントローラ側では、ダウンロードした基本関連情報に基づき、実際のデバイスと通信したりして、空欄の情報を埋めて関連情報を自動的に生成するようにしたが、マニュアル操作で関連情報を作成しても良い。また、基本関連情報のときから、必要な情報を全て（一部はマニュアル操作の場合もある）格納するようにすることもできる。その場合には、情報の内容としては、基本関連情報と関連情報は等価となる。なお、「関連情報」は、実施の形態ではオブジェクトデータベース１３に対応し、「基本関連情報」は

、オブジェクトデータベース２２に対応する。

【００３８】

さらに本発明では、制御プログラムがデバイスに対してアクセスするに際し、直接アドレス等を指定するのではなく、関連情報を参照してアクセス先を特定するので、上記したコントローラの使用方法、処理方法などとして以下に示す各種の方法が提供される。

【００３９】

すなわち、コントローラに接続されるデバイスを管理するための制御プログラムと、前記制御プログラムと前記デバイスを関連付けた関連情報とを対にし、あるコントローラで利用している制御プログラムを別のコントローラで再利用する場合には、その制御プログラムとともに、対となった前記関連情報も前記別のコントローラに実装する。次いで、前記関連情報中のデバイスのアクセス先に関する情報を前記別のコントローラにおけるデバイスのアクセス先に合わせて修正するようにした制御プログラムの再利用方法が提供できる。この場合に、制御プログラムに対する修正をすることなく、関連情報を修正するだけで別のコントローラに実装することができ、再利用が簡単に行える。

【００４０】

また、コントローラに接続されるデバイスを管理するための制御プログラムと、前記制御プログラムと前記デバイスを関連付けた関連情報を記憶する関連情報記憶手段とを備え、前記制御プログラムが前記デバイスにアクセスするに際し、前記関連情報記憶手段に格納された関連情報を参照することによりアクセス先を特定し、実行するコントローラにおける関連情報の管理方法としては、前記関連情報に、前記デバイスの設定パラメータを含む場合、コントローラの起動時に前記関連情報に記憶された前記設定パラメータを前記デバイスにダウンロードする。そして、前記コントローラの終了時に前記デバイスに設定されている設定パラメータをアップロードして記憶保持するとよい。この発明は、第２の実施の形態で実現されている。

【００４１】

さらに、コントローラに接続されるデバイスを管理するための制御プログラム

と、前記制御プログラムと前記デバイスを関連付けた関連情報を記憶する関連情報記憶手段とを備え、前記制御プログラムが前記デバイスにアクセスするに際し、前記関連情報記憶手段に格納された関連情報を参照することによりアクセス先を特定し、実行するコントローラ内のコントローラメモリ上に、デバイスのステータス情報を記憶する領域をデバイスごとに設定し、前記コントローラは、前記領域を監視し、異常を検知した場合に、前記関連情報記憶手段をアクセスして異常を発生しているデバイスの関連情報を取得し、所定の情報を外部周辺機器に通知する異常監視方法を実現することもできる。この発明は、第３の実施の形態に対応する。

【００４２】

さらにまた、請求項１から５のいずれか１項に記載のコントローラと、デバイスが接続されたネットワークに情報処理装置を接続し、前記情報処理装置は、前記コントローラに実装された前記制御プログラム及び関連情報と同等のものを備えるとともに、前記コントローラと前記デバイス間の通信状態を参照する機能を備え、前記参照した通信状態に従い、前記情報処理装置内の制御プログラムを実行して、前記コントローラと前記デバイスからなるシステム動作状況を監視等の情報処理を行い、その結果を前記コントローラに通知する情報処理方法を実施することもできる。この発明は、第４の実施の形態に対応する。

【００４３】

この発明の以上説明した構成要素は可能な限り組み合わせることができる。この発明による装置を構成する各手段を専用のハードウェア回路によって実現することができるし、プログラムされたコンピュータによって実現することもできる。

【００４４】

【発明の実施の形態】

図３は、本発明が適用されるシステムの一例を示している。また、図４は、ツール２０の内部構造を示している。図３に示すように、本システムは、ＰＬＣ等のコントローラ１０とツール２０とにより構成され、コントローラ１０には、直接またはネットワークを介してデバイス３０が接続されている。コントローラ１

０は、デバイス３０の動作を管理する制御プログラム１１を有する。また、デバイス３０は、割り付けられたコントローラメモリ（Ｉ／Ｏメモリ）１２の所定エリアに対し、データを読み込み或いは書き込みを行う。これらは、通常のコントローラとしての機能と同様であるのでその詳細な説明を省略する。

【００４５】

ここで、本発明では、制御プログラム１１とデバイス３０とを一体にした制御オブジェクトという単位で取り扱いを可能とし、係る取り扱いを行うための必要な情報をオブジェクトデータベース１３に格納する。

【００４６】

すなわち、本形態では、デバイス３０を特定するのに際し、オブジェクト名（デバイス名）を用いて識別するようにし、プログラム中でデバイスを読み出す場合、従来はアドレスを直接入力するようにしていたが、本形態ではデバイス名で定義するようにした。そして、デバイス名で特定されるデバイスが実際にどこに接続され存在するかの情報を別途用意するようにした。そして、具体的にどこに接続されているか等の情報は別途管理するようにした。これにより、再利用のために制御プログラム１１とデバイス３０を別のコントローラに接続した際に、アドレス（メモリアドレス，通信アドレス等）が異なった場合でも、別途管理しているデバイス名とアドレスの関係を修正するだけで済み、制御プログラムの内容の修正までは不要となる。

【００４７】

係る処理を実現するため、オブジェクトデータベース１３には、制御オブジェクトを構成するデバイス情報とデバイスをオブジェクト名で識別するための情報を保持するために、「制御オブジェクト名」と「制御オブジェクトを構成するデバイス情報」を格納している。ここで、デバイス情報としては、バス或いはネットワーク上におけるデバイスのアドレス（以下、通信アドレス）である。すなわち、デバイスがＰＬＣ等の内部状態を保持できる場合は、ユーザがＰＬＣ等に対して定義したデバイス名とそのデバイス名がＰＬＣ等のデバイスのどのエリアに保存されているかというアドレス情報がある。更に、デバイスが単純なデジタルＩ／Ｏのように内部状態を保持できない場合は、ユーザが定義するデバイス名に

加え、デバイスの製造メーカが定義するデバイスのシリアル番号なども併せて格納するようにしている。さらにまた、ユーザが定義した制御オブジェクトのインタフェース情報（以下、オブジェクトインタフェース情報：サービス名、サービスに必要なデータの型情報、オブジェクトの属性名、及びその型情報）を保持する仕組みを持つ。

【００４８】

よって、オブジェクトデータベース１３により、制御オブジェクトがどのようなデバイスから構成されているかということと、デバイスをオブジェクト名という論理名で識別することが可能となる。

【００４９】

上記各情報は、ツール２０により作成されたものをコントローラ１０にダウンロードすることにより実装される。また、コントローラ１０にダウンロードされる際に必要なデータの修正が行われる。以下、データの生成順に従い、説明する。

【００５０】

まず、ツール２０は、図５に示すフローチャートを実現する機能を持ち、これによりオブジェクトデータベースに格納する情報を生成する。すなわち、ツール２０は、パソコンその他のコンピュータなどにより構成される。そして、図６に示すように、そのモニタ画面Ｍに、接続可能なデバイスのリスト（Ｈ／Ｗ　Ｃａｔａｉｏｇ）が表示されるリスト領域（デバイスカタログＶｉｅｗ）Ｒ１と、実際にグラフィック画面で１つの制御オブジェクトを組むための作業領域Ｒ２と、その作業領域Ｒ２の情報に作成中の制御オブジェクト名を登録する領域Ｒ３を有している。作業者（開発者）は、このモニタ画面Ｍを見ながら各種操作をすることになる。

【００５１】

具体的には、まず最初にオブジェクト定義処理部２１が、制御オブジェクト名を定義する（ＳＴ１）。つまり、ツール２０のポインティングデバイスを操作してカーソルを領域Ｒ３にセットし、キーボードなどの入力装置を操作して制御オブジェクト名を入力する。オブジェクト定義処理部２１は、係る入力を受けて、

オブジェクト名を登録する。図６では、制御オブジェクト名として、「ネジ締めロボット」が登録されていることを示している。

【００５２】

次に、制御オブジェクトの構成デバイスを定義する（ＳＴ２）。具体的には、図６に示すモニタ画面Ｍのリスト領域Ｒ１内から使用するデバイス（クラス）を選択し、右側の作業領域Ｒ２にドラッグ＆ドロップして貼り付ける。デバイスが複数存在する場合には、この処理を複数回繰り返すことにより、制御オブジェクトに必要なデバイス構成を定義する（図７には、３つのデバイスＡ，Ｂ，Ｃが選択されたことを示している）。

【００５３】

そして、上記のようにして選択された各デバイスの内容を設定する。具体的には、設定するデバイスをダブルクリックすると、図６，図７に示すように、設定画面Ｇが表示される。図示の便宜上モニタ画面Ｍの外側に記載しているが、実際には作業領域Ｒ２の情報に重ねて描画される。そして、この設定画面Ｇ上で、デバイス名（図では「Ｃ」），通信アドレス，ＩＮデータサイズ，ＯＵＴデータサイズ，通信方法等のネットワーク通信のコンフィグレーションに必要な情報を入力する。すると、オブジェクト定義処理部２１では、その入力されたデータに基づきデバイスの設定情報をＸＭＬ，Ｉｎｉファイル等の所定のファイル形式でオブジェクトデータベース２２に格納する。

【００５４】

なお、各デバイスについての基本機能（デバイス名，ＩＮデータサイズ，ＯＵＴデータサイズ）は、初期値としてデータベースに登録されているので、上記のように、ダブルクリックされると、その指定されたデバイスについての初期値を読み出し、設定画面Ｇに表示する。よって、操作者は、空欄に所定のデータを入力することになる。そして、上記の設定により、オブジェクトデータベース２２に格納された内容の一例としては、図８に示すようになる。また、係る処理を繰り返し行うことにより、「ネジ締めロボット」を構成する各デバイスＡ，Ｂ，Ｃについての情報は、図９に示すような内容で登録される。

【００５５】

なお、デバイス番号（ＤｅｖｉｃｅＮｕｎ）や、ノード番号等は、空いてる番号や昇順方式などにより自動的に振り付けるようにしても良い。なおまた、この時点では、デバイスのシリアル番号や ＩＮデータの先頭アドレス（ＩＮａｄｒ０）やＯＵＴデータの先頭アドレス（ＯＵＴａｄｒ０）は、格納されない。

【００５６】

次に、オブジェクトインタフェース定義処理部２４が、制御オブジェクトのインタフェースを定義する（ＳＴ３）。実際には、この処理部がＩＮデータサイズやＯＵＴデータサイズを初期値としてモニタ画面に表示し、それを見ながら開発者が定義を入力するので、それを受けてユーザが定義した情報をオブジェクトデータベース２２に格納する。図１０（デバイスＣについて）では、デバイスのＩＮ領域の下位１ｂｙｔｅをＩｎ＿ｐｒａｍ１，上位１ｂｙｔｅをＩｎ＿ｐｒａｍ２とユーザが定義し、ＯＵＴ領域の下位１ｂｙｔｅをＯｕｔ＿ｐｒａｍ１，上位１ｂｙｔｅをＯｕｔ＿ｐｒａｍ２とユーザが定義した情報がオブジェクトデータベース２２に格納されたことを示している。

【００５７】

次いで、デバイスに対する制御プログラムを記述する（ＳＴ４）。すなわち、制御プログラム作成部２３により別途汎用言語で作成された制御プログラムを、制御オブジェクトの１つのメソッドとして定義され、オブジェクトデータベース２２に選択された制御プログラムが、制御オブジェクトのサービスとして登録される。

【００５８】

一例を示すと、例えば図１１に示すような「Ａｄｄ＿Ｖａｌ」という制御プログラムが作成され、登録されているものとする。この場合に、制御オブジェクトの１つのメソッドとして「Ａｄｄ＿Ｖａｌ」が定義され、オブジェクトデータベース２２に「Ａｄｄ＿Ｖａｌ」という制御プログラムが「ネジ締めロボット」というオブジェクトのサービスとして登録される。つまり、オブジェクトデータベース２２には、ユーザが定義した制御オブジェクトのインタフェース情報（サービス名，サービスに必要なデータの型情報，オブジェクトの属性名，及びその型情報）が登録されることになる。これにより、図８に示した記憶内容が、図１２

に示すようになる。

【００５９】

また、制御プログラムは、Ｊａｖａ，ＦＢ，ＶＢ等で記述される。そして、Ｊａｖａで記述されている場合には、「ネジ締めロボット」がＪａｖａの１つのクラスとなる。またＩＥＣ１１３１で記述されている場合、「ネジ締めロボット」が１つのＦＢとなる。

【００６０】

そして、制御オブジェクト管理部２５が、制御オブジェクトをデータベース２６に登録する（ＳＴ５）。登録の際には、オブジェクトデータベース２２を参照して、制御オブジェクトを生成する。ここでの生成とは、Ｊａｖａの場合、１つのＪａｖａクラスのコード、ＦＢならＦＢのコードを生成することを示す。図１３では、オブジェクトデータベースからＪａｖａのクラスが生成された場合の例を示している。生成されたネジ締めロボットＪａｖａクラスの属性に、「ネジ締めロボット」という情報が格納される。これにより、オブジェクトデータベース２２と制御プログラムが関連付けられる。

【００６１】

なお、説明が前後するが、ここで関連付けられて登録する制御プログラムは、予めクラス作成処理部２７にて制御クラスが生成されている。つまり、図４に示すように、このクラス作成処理部２７は、オブジェクトデータベース２２を参照し、ユーザが汎用言語で作成した制御プログラム２８から汎用言語のクラスを生成するもので、生成されたクラスの属性として、オブジェクトデータベースに格納されている制御オブジェクト名が格納される。図１３の例では、属性として「ネジ締めロボット」という制御オブジェクト名が格納された制御プログラム２８′が生成される。これにより、制御プログラム（上記クラスのメソッド群）２８′と、オブジェクトデータベース２２が関連付けられ、制御プログラムとデバイスを対として制御オブジェクトとして扱うことが可能となる。

【００６２】

また、ユーザが作成する制御オブジェクトのサービス以外に以下のようなシステムで自動的に生成されるサービスが存在する。すなわち、「Ｇｅｔ＿Ｐｒｏｆ

ｉｌｅ」は、制御オブジェクトのプロファイル情報を獲得する。例えば制御オブジェクトを構成するデバイスの通信アドレスを獲得する処理を実行する。

【００６３】

また、「Ｓｅｎｄ＿Ｍｅｓｓａｇｅ／Ｒｅｃｅｉｖｅ＿Ｍｅｓｓａｇｅ」は、制御オブジェクトを構成するデバイスに対してメッセージを送信／受信する際のインタフェースである。ＤｅｉｖｃｅＮｅｔ（登録商標）の場合、「デバイス名，クラスＩＤ，インスタンスＩＤ，アトリビュートＩＤ」などのパラメータをもつ。このサービスが呼ばれた場合、このサービスのパラメータであるデバイス名をキーにオブジェクトデータベース２２から通信アドレスを獲得し、パラメータを当該デバイスに対して送信する処理を実行する。これにより、制御オブジェクトにプログラムは、デバイスの識別子として通信アドレスを指定するのでなく、オブジェクト名でアクセスすることが可能となる。換言すると、通信アドレスは知らなくても、或いは、変更されていたとしても、オブジェクト名と通信アドレスの関連付け際しておくことにより、デバイスを呼び出すことができる。

【００６４】

さらに、「Ｓｅｔ＿Ａｔｔｒｉｂｕｔｅ／Ｇｅｔ＿Ａｔｔｒｉｂｕｔｅ」は、制御オブジェクトの属性に値を設定したり、制御オブジェクトの属性の値を獲得する処理を実行する。

【００６５】

また、制御オブジェクト管理部２５は、作成したデータベース２６に登録された制御プログラム２８′とオブジェクトデータベース２２を、コントローラ１０にダウンロードする（ＳＴ６，ＳＴ７）。制御オブジェクトを定義する開発環境がコントローラ内に存在する場合は、同一装置の決められた場所（メモリの番地）に配置することを意味する。

【００６６】

一方、コントローラ１０は、ツール２０からダウンロードされた情報（オブジェクトデータベース１３に格納）に従い、コントローラシステムソフト１４が、以下の処理を実行する。すなわち、図１４に示すように、制御オブジェクト割付処理部１４ａは、オブジェクトデータベース１３を参照し、制御オブジェクトの

構成デバイスとＩＮサイズ，ＯＵＴサイズを獲得し（①）、そしてコントローラメモリ１２の空き領域にメモリマップを行う（②）。

【００６７】

すなわち、オブジェクトデータベース１３に格納された「ネジ締めロボット」についての情報が図９のようになっているとすると、デバイスＣについては、ＩＮサイズが「2」であるので、図１３に示すようにコントローラメモリ１２のあいている所定のアドレス（図示の例では「ＡＤ１」）から２バイト分の領域を確保することにより、割り付け（マッピング）を実行する。

【００６８】

そして、この割り付けに伴い、割り付けたコントローラメモリ１２上の先頭アドレス（図示の場合「ＡＤ１」）をオブジェクトデータベース１３に格納する（③）。

【００６９】

そして、この割付によりデバイスＣについての終了アドレスはわかっているので（先頭アドレス＋ＩＮデータサイズ）、次のデバイスＡについては、その終了アドレスの次のアドレスを先頭アドレスＡＤ３として割付を行い、その結果をオブジェクトデータベース１３に格納する。以下順に繰り返し処理をすることにより、全てのデバイスについてのＩＮデータとＯＵＴデータの割付が行え、図９に示す例では、図１６に示すようなデータが作成される。

【００７０】

次に、デバイスのシリアル番号を獲得し、オブジェクトデータベース１３へ登録する。すなわち、図１６に示すように、通信処理部１４ｂが、オブジェクトデータベース１３へアクセスし、処理対象のデバイスのノード番号とデバイス名等の情報を取得する。そして、その取得したノード番号に接続されたデバイスに対してデバイス名，シリアル番号などの問い合わせを行い、デバイスからの返答を待ち、デバイスから送られてきた情報が正しい（オブジェクトデータベース１３に登録されている情報と一致する）場合に、送られてきたシリアル番号を正式なものと判断し、オブジェクトデータベース１３の該当領域に登録する。

【００７１】

図１６に示す例では、ノード番号（＃８）に接続されたデバイスは、デバイス名（オブジェクト名）がＣで、シリアル番号が「ＳＮ－０１」であるので、デバイス名が一致するため、図１５に示すように、デバイス名Ｃのところに、シリアル番号「ＳＮ－０１」を格納する。係る処理を全てのデバイスに対して行い、図１５に示すようなネジ締めロボットについてのオブジェクトデータベースが完成する。

【００７２】

また、ＰＬＣのように装置内部にデータを保持できるものについては、デバイスと通信する際に、オブジェクトデータベースに格納されているデバイスのデバイス名をデバイスの決められたエリア（どこに書き込むかはオブジェクトデータベースに格納されている）に書き込む。またこの際に、すでにデバイスにデバイス名が書き込まれている場合は、マッチングをかける。もし一致しない場合は異常を外部周辺機器などに通知する。

【００７３】

これにより，オブジェクトデータベースとテバイスの関連付けができる。また、制御オブジェクト単位のＩ／Ｏコンフィグレーションが可能となる。つまり、制御プログラムがデバイスを読み出す場合、デバイス名でプログラムを組んでおくと、オブジェクトデータベース１３によりデバイス名とそのデバイスが接続されたノード番号や、そのデバイスについてのＩＮ／ＯＵＴデータを格納するメモリアドレスが関連付けられているので、その関連付けられた情報に基づきアクセスすることができる。

【００７４】

一方、制御オブジェクト（制御プログラム＋デバイス）の再利用をする場合には、以下の処理手順を実行することにより行われる。すなわち、ツール２０内の制御オブジェクト管理部２５が、制御プログラムとオブジェクトデータベースを管理しているデータベース２６から、再利用する制御プログラムとオブジェクトデータベースを取り出し、取り出した情報（制御プログラム＋オブジェクトデータベース）を再利用先のコントローラ１０に、ダウンロードする。なお、オブジェクト定義ツールでは、制御オブジェクトを指定してコントローラ１０にダウン

ロードすると、制御プログラムとオブジェクトデータベースが対でダウンロードされるようになっている。また、制御オブジェクトが複数の制御オブジェクトから構成される場合でも関係する制御オブジェクトの制御プログラム，オブジェクトデータベースもすべてダウンロードされる。また、制御プログラムと対になるデバイス３０をコントローラ１０に接続する。この接続は、使用状況に合わせてネットワーク，直接のいずれでも良い。

【００７５】

そして、再利用先のコントローラ１０における制御オブジェクト割付処理部１４ａと通信処理部１４ｂが、それぞれ以下のような処理を実行する機能を有する。まず、制御オブジェクト割付処理部１４ａは、図１７に示すように、すでにあるオブジェクトデータベース１３′を参照して空き領域を検出し、新たにダウンロードしたオブジェクトデータベースの各デバイスについてのコントローラメモリを割り付ける。つまり、すでに存在するオブジェクトデータベースの割付情報（ＩＮ／ＯＵＴサイズ、割付けアドレス）を獲得し（①）、再利用のためにダウンロードされた制御オブジェクトのオブジェクトデータベースから割付情報（ＩＮ／ＯＵＴサイズ）を獲得する（②）。そして、上記①の処理に基づき求めたコントローラの空き領域に、②の処理で得られた新たなデバイスについてマッピングする（③）。

【００７６】

次いで、そのマッピング情報（コントローラメモリ１２に割付けた先頭アドレス）をオブジェクトデータベース１３″に格納する（④）。これにより、新たにダウンロードしたデバイスに対するメモリ割付が完了する。そして、空き領域に行われるので、既存のオブジェクトデータベース１３′への影響はない。もっとも、この再利用に伴う割付処理に際し、必要に応じて既存のオブジェクトデータベース１３′に割り付けられているアドレスを変更してもかまわない。仮に係る変更をした場合でも、データが更新されるのはオブジェクトデータベース１３′内の先頭アドレスを格納する領域であり、係る既存のデバイスに対する制御プログラムは、デバイス名で呼び出すので影響を与えないからである。なお、上記した処理②から④は、基本的に新規の割付処理（図１４の①から③）と同様である

ので、その詳細な説明を省略する。

【００７７】

一方、図１８に示すように、通信処理部１４ｂは、既存のオブジェクトデータベース１３′からオブジェクト名とシリアル番号と通信アドレス（以下、登録デバイス構成）を獲得する（①）。次にネットワーク上のデバイスに対して何が接続されているのかを問い合わせる（②）。つまり、全てのノードに対し、オブジェクト名と通信アドレス（ノード番号）の回答を要求する。すると、係る要求を受けた各デバイスは、要求に応じた回答をするので、実際に接続されているデバイスのデバイス名（デバイス名を記憶できないデバイスは、シリアル番号）と通信アドレスの対応（以下、実デバイス構成）を獲得後、オブジェクトデータベースから獲得された登録デバイス構成と比較を行う。

【００７８】

これにより、実デバイス構成と登録デバイス構成があっているか否かを判断するとともに、あいている通信アドレスを認識する。既存の登録デバイス構成と異なるものがあれば、オブジェクトデータベースの記憶内容を実デバイス構成に併せて修正する。

【００７９】

そして、ダウンロードしてきたオブジェクトデータベースに登録されているデバイスの通信アドレスが、既存の登録デバイスに付与されている通信アドレスと重複しているか否かを判断し、重複していない場合には、そのままの通信アドレスを設定する。一方重複している場合には、あいている通信アドレスに自動的に割付を行い、そのアドレスをオブジェクトデータベースに格納する（③）

図１８に示したものの場合、すでに存在するデバイス（Ｄ，Ｅ，Ｆ）が通信アドレス＃８，＃３，＃１を使用しており、制御オブジェクト「ネジ締めロボット」の構成要素であるデバイス（Ａ，Ｂ，Ｃ）の通信アドレスと重複している。従って、例えばデバイスＣは、あいている通信アドレス＃２を自動的に割り付けられたとすると、図１８に示すように、ノード番号が「８」から「２」に更新される。他のデバイスに対しても同様である。

【００８０】

これにともない、デバイスに対するノード番号の設定も行う。このとき、係るデバイスの通信アドレスの設定が、ネットワーク経由で行えない場合（デバイスのディップスイッチ等のハードウェアスイッチで設定する）は、外部周辺機器に通信アドレスの再設定する必要があることを通知する。

【００８１】

これにより、制御オブジェクトを流用した場合でも関係ない制御オブジェクトには影響を与えない。また、通信アドレスが重複した場合でも自動割付或いは手動割付により、ユーザの負荷が減少される。また通信アドレスが変更された場合でも制御プログラム（デバイス名の変更はないため）には影響を与えない。

【００８２】

さらにまた、コントロールシステムを構築する場合に、あるデバイスがそろわないために、代用のデバイス（クラスは同じ）で対応し、インクリメンタル開発をする場合や、あるデバイスのメーカを変更したり、２つのデバイスを１つにする（８点デバイス２個を１６点デバイス１個にする）等のデバイスの変更が発生した場合、図１３の構成では、デバイスの情報は、全て関数コールで書き込み、参照となっていることから、制御オブジェクトのインタフェースを変更することなく，制御オブジェクトのメソッドを変更することなく、デバイスのＩ／Ｏコンフィグレーションの部分だけを実行すればよい。オブジェクト指向でいう、ポリモフィズムが達成される。これにより、コントロールシステム構築の際にインクリメンタル開発，設備変更が容易となる。

【００８３】

図１９は、本発明の第２の実施の形態を示している。本実施の形態では、上記した実施の形態を基本とし、さらに、設備の起動時にコントローラからデバイスに設定パラメータをダウンロードしたり、設備のシャットダウン時にデバイスの設定パラメータをアップロードする機能を設けている。

【００８４】

まず、コントローラの動作を決定するシステム設定を記憶する記憶部１５に、「電源ＯＮ時のデバイスパラメータのダウンロードの要否」並びに「電源ＯＦＦ時のデバイスパラメータのアップロードの要否」情報を記憶するようにする。そ

して、それぞれに対してＹｅｓ／Ｎｏの設定を行う。この設定は、例えば、コントローラ１０に制御オブジェクトをダウンロードするタイミングで一緒にダウンロードすることにより登録することができる。

【００８５】

そして、コントローラシステムソフト１４は、電源ＯＮ時に図２０に示すフローチャートを実行する機能と、電源ＯＦＦ時に図２１に示すフローチャートを実行する機能を有している。

【００８６】

まず、図２０に示すように、設備の電源がＯＮされ、設備内のコントローラも電源がＯＮされた場合、記憶部１５内に格納されたシステム設定を参照する（ＳＴ１０）。そして、上記システム設定のうち「電源ＯＮ時にデバイスパラメータをダウンロードする」が有効になっているか否かを判断する（ＳＴ１１）。そして、Ｎｏの場合には、そのまま処理を終了する。つまり、通常の制御動作に移行する。

【００８７】

一方、Ｙｅｓとなっていた場合には、コントローラ１０内に存在するオブジェクトデータベース１３の内容を参照し、オブジェクトデータベース１３から制御オブジェクトの構成要素であるデバイスの通信アドレス，設定パラメータを獲得する（ＳＴ１２，ＳＴ１３）。そして、獲得した通信アドレスのデバイスに対して、設定パラメータをダウンロードする（ＳＴ１４）。これにより、電源が投入される都度、デバイスの設定はその都度オブジェクトデータベース１３に登録された設定パラメータの内容に書き換えられる。

【００８８】

一方、設備のシャットダウン時の処理機能は、図２１に示すように、コントローラ１０の外部周辺機器からコントローラ１０に対してコントローラ１０のシャットダウン（コントローラのモード変更）指示をした場合、実行されている制御プログラムを停止し、コントローラシステムソフト１４は、記憶部１５に格納されているシステム設定を参照する（ＳＴ２１）。

【００８９】

そして、システム設定の「電源ＯＦＦ時にデバイスパラメータをダウンロードする」が有効（Ｙｅｓ）であるか否かを判断する（ＳＴ２２）。Ｎｏの場合には、そのまま処理を終了、つまりシャットダウンする。

【００９０】

一方、コントローラ１０内に存在するオブジェクトデータベース１３の内容を参照し、デバイス情報（通信アドレス）を取得する（ＳＴ２３，ＳＴ２４）。そして、取得した通信アドレスのデバイスに対して、デバイスの設定パラメータを読み込む（アップロードロードする）とともに、その読み込んだ設定パラメータを該当するオブジェクトデータベース１３の領域に格納する（ＳＴ２５，ＳＴ２６）。これにより、コントローラが起動した後で、現場調整で設定パラメータが変更された場合には、その変更された内容がオブジェクトデータベース１３に登録される。

【００９１】

これにより、例えば、両方の設定をＯＮにしておくと、電源が投入されると、常に前回シャットダウンしたときの情報で起動することができる。これにより、調整が以降も生きるので有意義となる。また、デバイスの設定パラメータを現場調整したときでも、調整された設定パラメータが制御オブジェクトの属性としてオブジェクトデータベースに格納され、保持されるので、実際のデバイスの設定パラメータと、オブジェクトデータベースの内容が整合された状態が保証され、設定パラメータの管理が容易に行える。また、再利用を考えた場合も、実際のデバイスの設定パラメータの値と整合がとれた状態で再利用されるので好ましい。なお、電源ＯＮ時のみＹｅｓにしておくと、前回の動作中に行った調整結果は保持されないので、調整前の状態に戻る。よって、一時的な調整などの場合に適する。

【００９２】

次に、具体的な処理機能について説明する。まず、図１９に示す例では、制御オブジェクトが「ネジ締めロボット」と「パーツフィーダ」の２つ存在することを示している。また、「ネジ締めロボット制御オブジェクトは、３つのデバイスＡ，Ｂ，Ｃから構成されており、デバイスＡの通信アドレスとデバイスパラメー

タがそれぞれ「３」、「５」であることを示している。また上記の通信アドレス、デバイスパラメータ以外に、設定パラメータをデバイスのどこに格納するのかといったアドレス情報（以下、設定パラメータアドレス情報）が存在する。ＤｅｖｉｃｅＮｅｔ（登録商標）の場合、クラスＩＤ，インスタンスＩＤといった情報が格納される。またデバイスＢの通信アドレス、デバイスパラメータがそれぞれ「１」，「２」であることを示している。またデバイスＣの通信アドレス、デバイスパラメータがそれぞれ「８」，「１０」である。

【００９３】

次に「パーツフィーダ」制御オブジェクトは、Ｘという１つのデバイスで構成されており、通信アドレス，デバイスパラメータがそれぞれ１０，１００である。

【００９４】

この場合において、電源ＯＮ時は、各デバイスＢ，Ａ，Ｃ，Ｘに対して、それぞれ「３，５，１０，１００」がダウンロードされる（図１９中実線）。そして、デバイスはコントローラよりダウンロードされた設定パラメータで動作する。なお、コントローラ１０は制御プログラムの実行を開始する。

【００９５】

一方、デバイスＢについての設定パラメータは、使用中に変更（３→２）されている。。そして、「電源ＯＦＦ時にデバイスパラメータをダウンロードする」が有効（Ｙｅｓ）であるので、シャットダウンの際には、各デバイスの設定パラメータ、つまり、デバイスＢ，Ａ，Ｃ，Ｘに対して、それぞれ「２，５，１０，１００」が返される。これにより、デバイスＢの設定パラメータの値として３がダウンロードされた後、現場調整で２に変更された情報が保持される。

【００９６】

また、上記したように電源ＯＮ時に設定パラメータのダウンロードを行う機能を利用することにより、以下のような新たな機能が実現できる。係る場合において、コントローラシステムソフト１４が各デバイスに対してダウンロードするに際し、デバイス名で通信先を特定するようにする。

【００９７】

すなわち、例えばデバイスが故障した場合、一旦電源をＯＦＦにし、新たなデバイスに交換後、電源をＯＮにする。このとき、交換後のデバイスは、交換前のデバイスとデバイス名は同じでもシリアル番号が異なる。また、通信アドレスは、ディップスイッチなどのハード的なスイッチでの設定を行わない限り、未設定となる。

【００９８】

そこで、上記したようにデバイス名で問い合わせをすることにより、交換後のデバイスもコントローラ１０と通信ができる。もちろん、既存のデバイスも通信ができることは言うまでもない。そして、デバイスは、デバイス名とともにシリアル番号も通知するようにする。すると、デバイスをデバイス名とシリアル番号（オブジェクトデータベースに格納）で管理しているので、コントローラは、起動時にオブジェクトデータベースからデバイスのシリアル番号を獲得し、交換されたデバイスのシリアル番号とマッチングをかける。すなわち、デバイス名が一致するデバイスのシリアル番号の異同をチェックする。

【００９９】

そして、一致していれば交換されておらず元のデバイスが接続されていることが確認できる。また、シリアル番号が異なる場合には、コントローラ１０（コントローラシステムソフト１４）は外部周辺機器に異なるシリアル番号のデバイスを制御オブジェクトのデバイスとして認めるかを問うメッセージを通知する。そして、外部周辺機器で、上記内容にＹｅｓというメッセージをコントローラに送信すると、コントローラ１０のコントローラシステムソフト１４は、新たなシリアル番号をオブジェクトデータベースに書き込む。

【０１００】

また、故障に伴い交換したデバイスに対しては、オブジェクトデータベース１３に格納された故障前のデバイスについての設定パラメータをダウンロードする。これにより、自動的に設定が必要なデバイスの抽出から、実際にダウンロードするまでの処理が行える。もちろん、このように自動的に行うのではなく、外部装置から交換したデバイス情報（デバイス名など）を与え、対応する交換前のデバイスについての設定パラメータを交換後のデバイスに登録したり、交換後のデ

バイスのシリアル番号のオブジェクトデータベース１３への登録などを行うようにしても良い。

【０１０１】

図２２は、第３の実施の形態を示している。本実施の形態では、デバイスに異常が発生した場合に、外部周辺機器に異常を通知する機能を持たせたものである。

【０１０２】

すなわち、コントローラメモリ１２の所定領域に、制御オブジェクトのデバイスのステータス（以下、デバイスステータス情報）を割り付ける。このデバイスステータス情報は、通信アドレスが異常なのか正常なのかを表す通信アドレス毎のステータス情報と、通信アドレスのデバイスが異常であった場合の異常原因を格納する領域を用意している。そして、デバイス３０は、自己の状態、つまり正常／異常情報とその異常原因を上記した割り付けられたコントローラメモリ１２内の所定エリアに格納する。

【０１０３】

上記の前提において、図３にも示したコントローラシステムソフト１４内に設けられたデバイス監視処理部１４ｃが、以下の処理を実行する。すなわち、図２３のフローチャートに示すように、コントローラ１０が実施する通常のスキャン処理に同期して、コントローラメモリ１２に格納されたデバイスステータス情報を監視する。すなわち、コントローラ１０は、例えば、ＩＮリフレッシュ→命令実行→ＯＵＴリフレッシュ→通信処理を１サイクルとして繰り返し実行するため、その１サイクルごとに（図示の例では、通信処理を実行後）１度、デバイスの監視処理を行う。

【０１０４】

このデバイスの監視処理は、具体的には、デバイスステータス情報を参照して、異常が発生している通信アドレスがないかを判定する。全ての通信アドレスのステータスに異常がない場合には、そのサイクルにおける処理を終了する。また、異常があった場合には、その異常が発生した通信アドレスについて割り付けられている異常原因情報を参照する。

【０１０５】

次いで、オブジェクトデータベース１３を参照して、異常が発生している通信アドレスがどの制御オブジェクトに該当するのかを検索する。そして、コントローラ１０（デバイス監視処理部１４ｃ）は、「異常が発生した制御オブジェクト名，異常が発生したデバイス名，異常が発生したデバイスの通信アドレス並びに異常原因」を、外部周辺機器、或いはコントローラのメモリにロギングする。

【０１０６】

なお、異常原因の登録の仕方としては異常コードとしてもよいし、メッセージとしてもよい。前者の場合には、記憶する際のメモリ容量が小さく済むが、ユーザが異常コードをもとに、帳票を参照して異常原因を調べる作業が発生するので煩雑となる。一方、後者の場合には、メッセージである場合は、オブジェクトデータベースのサイズが大きくなるというデメリットがある反面、異常原因を人間が直接わかる文字情報で表現しているため、その文字情報を表示したり、その文字情報を音声情報に変換してユーザに伝えるというようにユーザが理解しやすいように伝えることが可能となる。

【０１０７】

何れの場合も、故障したデバイスがどの制御オブジェクトに該当するか等の情報を知ることができるので、異常に対する保守作業の効率化が図れる。さらに、異常原因についても、オブジェクトメッセージに異常原因，保守情報を格納しておくことにより、保守作業がさらに向上する。さらに、オブジェクトデータベースに異常原因，保守情報などを記述しておくと、係る情報はコントローラ１０側に記憶されるので、保守のためにコントローラに接続するパソコン側にメッセージデータベースがなくても済むので、どのパソコンからコントローラにアクセスしても異常原因等を把握することができる。

【０１０８】

図２４は、本発明の第４の実施の形態を示している。同図に示すように、コントローラ１０並びにデバイス３０がＤｅｖｉｃｅＮｅｔ（登録商標）等のネットワークで接続されている。そして、本形態では、ＨＭＩ等の情報処理機器４０もＤｅｖｉｃｅＮｅｔ（登録商標）に接続されている。但し、この情報処理機器４０

はＤｅｖｉｃｅＮｅｔの１つのノードとして加入せずに、ＤｅｖｉｃｅＮｅｔ上を流れるメッセージを参照する機能を持つ（プロトコルアナライザと同じ）。

【０１０９】

このようにすると、コントローラ１０とデバイス３０間の通信サイクルに影響を与えないで、どのノード（通信アドレス）がどういうデータを通信しているかをこの情報機器が把握することが可能となる。

【０１１０】

この情報処理機４０内にも、第１の実施の形態で説明したものと同様の手順により、デバイスＡ，Ｂ，Ｃに対するオブジェクトを定義し（Ａ，Ｂ，Ｃ）、情報処理機器に制御プログラムとオブジェクトデータベースをダウンロードし、制御オブジェクトのＩ／Ｏコンフィグレーションを行う。そして、実際のシステム稼動時は、コントローラ１０は、デバイスＡ，Ｂ，Ｃの制御を行う。このとき、情報処理機器４０の中の制御プログラム（ここでは、情報処理目的で使われる）は、ＤｅｖｉｃｅＮｅｔ上を流れるメッセージを参照することにより、情報を収集し、収集したデバイスＡ，Ｂ，Ｃの情報をもとに演算処理をする。そして、その結果をコントローラ１０にフィードバックする。

【０１１１】

このようにすることにより、コントローラ１０が制御を行っている場合は、コントローラ１０の制御性能に影響を与えない形で、コントローラ１０が管理するデバイス３０の状態を情報処理機器４０からオブジェクト名という論理名でモニタすることが可能となる。そして、コントローラ１０が制御を行っていない場合は、情報処理機器４０からデバイス３０に対してオブジェクトという論理名でデバイスにデータを書き込むことも可能となる。

【０１１２】

このとき、デバイスのノード番号が制御プログラムの開発者の都合で変更された場合でも情報処理機器の制御オブジェクトにアクセスするプログラムに影響を与えない。また、すでにコントロールシステムが構築されていてコントローラの制御プログラムがオブジェクト化されていないような場合でも、あとから情報処理機器４０をネットワークに接続することにより、コントローラ１０が制御する

デバイス３０をオブジェクト名と論理名でアクセスすることが可能となる。従って、情報処理を記述する開発者は、オブジェクトベースの情報処理プログラミングが容易となり、開発生産性が向上する。

【０１１３】

さらにまた、コントローラ１０は、制御（性能も含め）に適した言語、プログラミングスタイル、デバイスへのアクセスの方法で開発を行い、情報処理は、上記コントローラがアクセスするデバイスに対してオブジェクト名という情報処理に適した論理名で情報プログラムが開発可能となるし、お互いの制御プログラム開発、情報プログラム開発が人，時期ともに分離可能となる。

【０１１４】

上記したように、デバイスとそれにアクセスする制御プログラムをオブジェクトデータベースという手段を用いて、１つの制御オブジェクトとして扱えるようにした。従って、一度開発した制御オブジェクト、すなわち「ハード＋ソフト一体」で再利用することが可能となる。これにより、生産システムにおいて、負荷変動、多品種少量生産、割込み的な生産要求にともなう段取り替えに要する時間が短縮される。

【０１１５】

制御オブジェクト単位で、コントローラへのメモリ割付を管理するため、再利用の場合でも、他の制御オブジェクトへの影響がなく、かつ再利用する制御オブジェクトを利用するプログラムへの影響がない。制御プログラムとオブジェクトデータベースを一体で管理するため、制御オブジェクトの流用が容易となる。

【０１１６】

また、再利用だけでなく、新たな制御オブジェクトを追加する場合でも他の制御オブジェクトに影響を与えないため、設備変更を伴う生産システムの改善が容易となる。

【０１１７】

複数のデバイスをグルーピングして制御オブジェクトとして扱ったり、複数の制御オブジェクトを１つの制御オブジェクトとして扱えるため、様々な粒度の制御オブジェクトをユーザが定義可能である。これにより、様々な再利用要求に対

応することが可能となる。

【０１１８】

デバイスの設定パラメータを制御オブジェクトの属性として扱い、設定パラメータをデバイスにダウンロード，アップロードをする仕組みを持たせるようにした場合には、デバイスの故障時の保守作業に要する時間が短縮される。

【０１１９】

デバイスが故障した場合にそのデバイスに対する制御オブジェクト名を特定する手段を持つことにより、どういう制御単位で故障が発生したかという意味付けが可能となり、保守作業が容易となる。

【０１２０】

デバイスに対して通信アドレスでなく、論理名でアクセスするため、通信アドレスの変更があった場合でも、制御オブジェクトを呼出しているプログラムへの影響がない。

【０１２１】

デバイスへのアクセスは、直接コントローラのメモリを参照していないため、制御オブジェクトのデバイスが変更された場合でもオブジェクトのインタフェース、及びオブジェクトのサービスの処理に影響を与えない。従って、コントロールシステムのインクリメンタル開発、及びデバイスの変更に柔軟に対応することが可能となる。

【０１２２】

【発明の効果】

以上のように、この発明では、制御プログラムとデバイスを関連付けた関連情報（オブジェクトデータベース）を用い、制御プログラム等がデバイスにアクセスするに際し、関連情報を参照することによりアクセス先を特定し、実行するようにしたため、制御プログラムとデバイスを一体でオブジェクト（制御オブジェクト）として扱うことができ、再利用が容易かつ確実に行え、再利用時に関係の無い制御オブジェクトへ影響を与えることが無くなる。

【図面の簡単な説明】

【図１】

従来例を示す図である。

【図２】

従来例を示す図である。

【図３】

本発明が適用されるシステム全体の一例を示す図である。

【図４】

ツールの内部構造を示す図である。

【図５】

ツールで行われる初期設定の機能を説明するフローチャートである。

【図６】

初期設定を説明する処理画面の一例を示す図である。

【図７】

初期設定を説明する処理画面の一例を示す図である。

【図８】

オブジェクトデータベースの一例を示す図である。

【図９】

オブジェクトデータベースの一例を示す図である。

【図１０】

オブジェクトデータベースの一例を示す図である。

【図１１】

制御プログラムの一例を示す図である。

【図１２】

オブジェクトデータベースの一例を示す図である。

【図１３】

クラス作成処理部２７の機能を説明する図である。

【図１４】

制御オブジェクト割付処理部の機能を説明する図である。

【図１５】

制御オブジェクト割付処理部により生成されたオブジェクトデータベースの一

例を示す図である。

【図１６】

通信処理部の機能を説明する図である。

【図１７】

再利用時の制御オブジェクト割付処理部の機能を説明する図である。

【図１８】

再利用時の通信処理部の機能を説明する図である。

【図１９】

本発明の第２の実施の形態を示す図である。

【図２０】

システムソフトの機能を説明するフローチャートである。

【図２１】

システムソフトの機能を説明するフローチャートである。

【図２２】

本発明の第３の実施の形態を示す図である。

【図２３】

作用を説明するフローチャートである。

【図２４】

本発明の第４の実施の形態を示す図である。

【符号の説明】

１０　コントローラ

１１　制御プログラム

１２　コントローラメモリ

１３　オブジェクトデータベース

１４　コントローラシステムソフト

１４ａ　制御オブジェクト割付処理部

１４ｂ　通信処理部

１４ｃ　デバイス監視処理部

２０　ツール

２１　オブジェクト定義処理部

２２　オブジェクトデータベース

２３　制御プログラム作成部

２４　オブジェクトインタフェース定義処理部

２５　制御オブジェクト管理部

２６　データベース

２７　クラス作成処理部

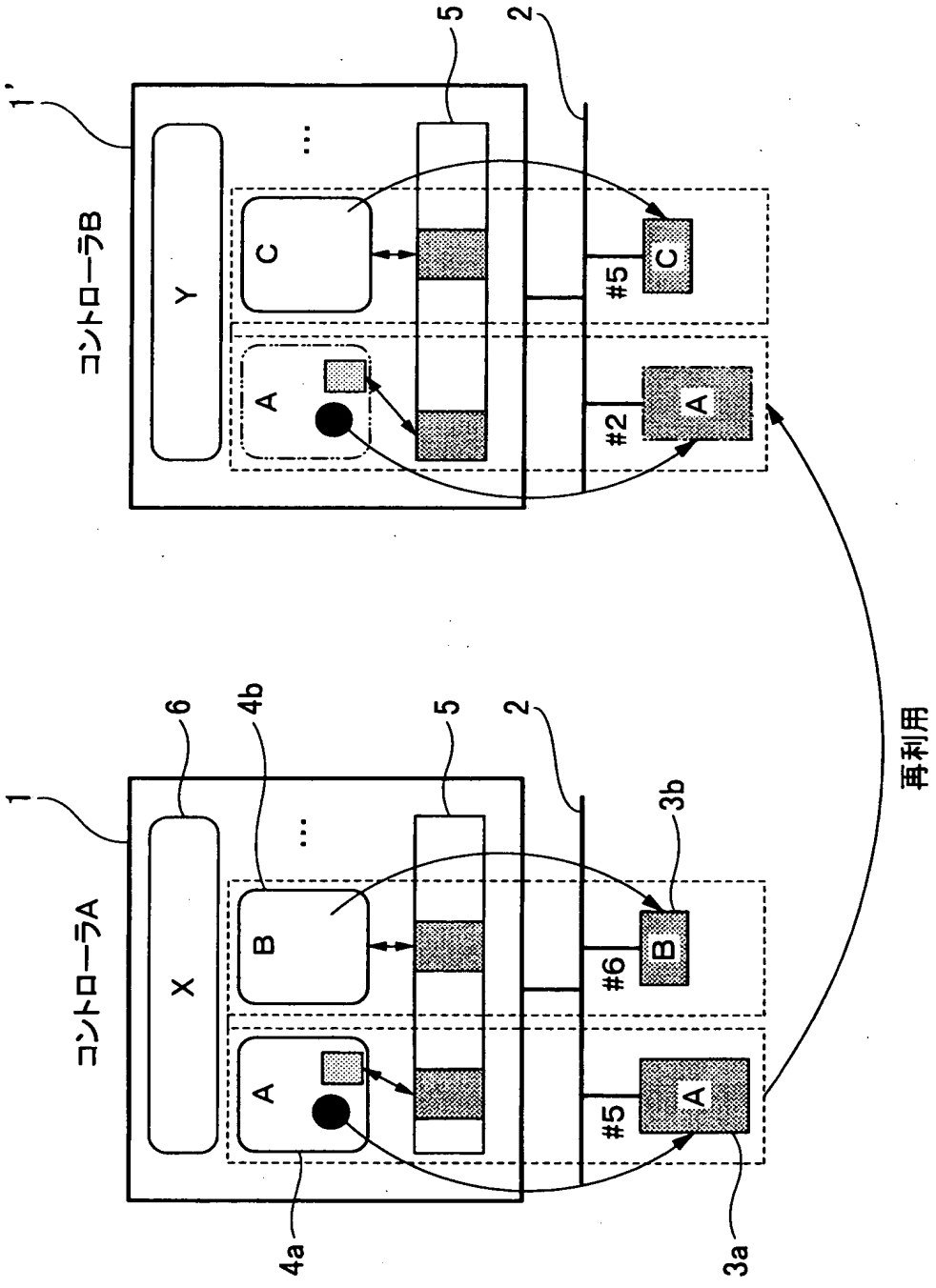
２８，２８′　制御プログラム

３０　デバイス

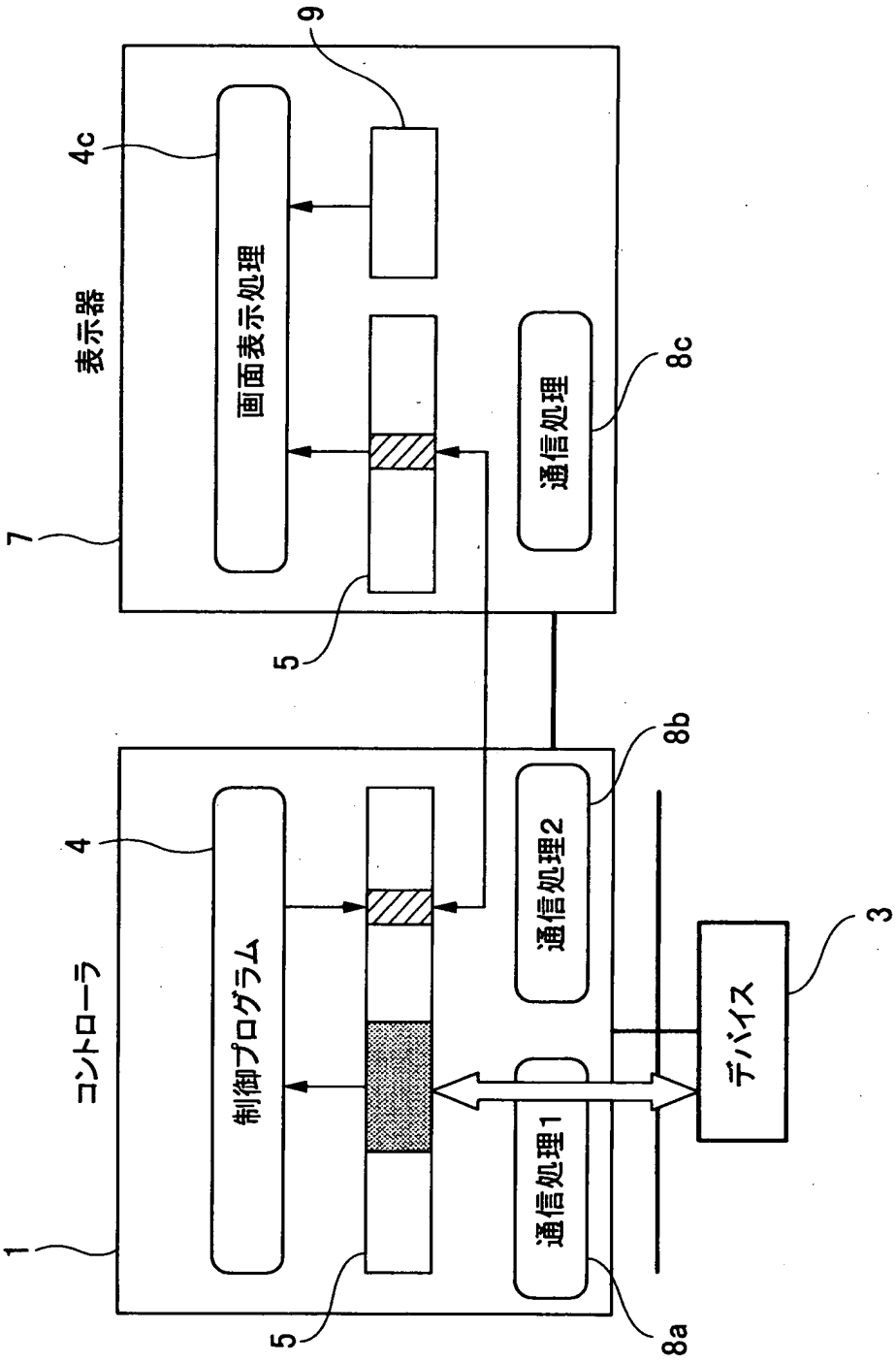
４０　情報処理機器

【書類名】　図面

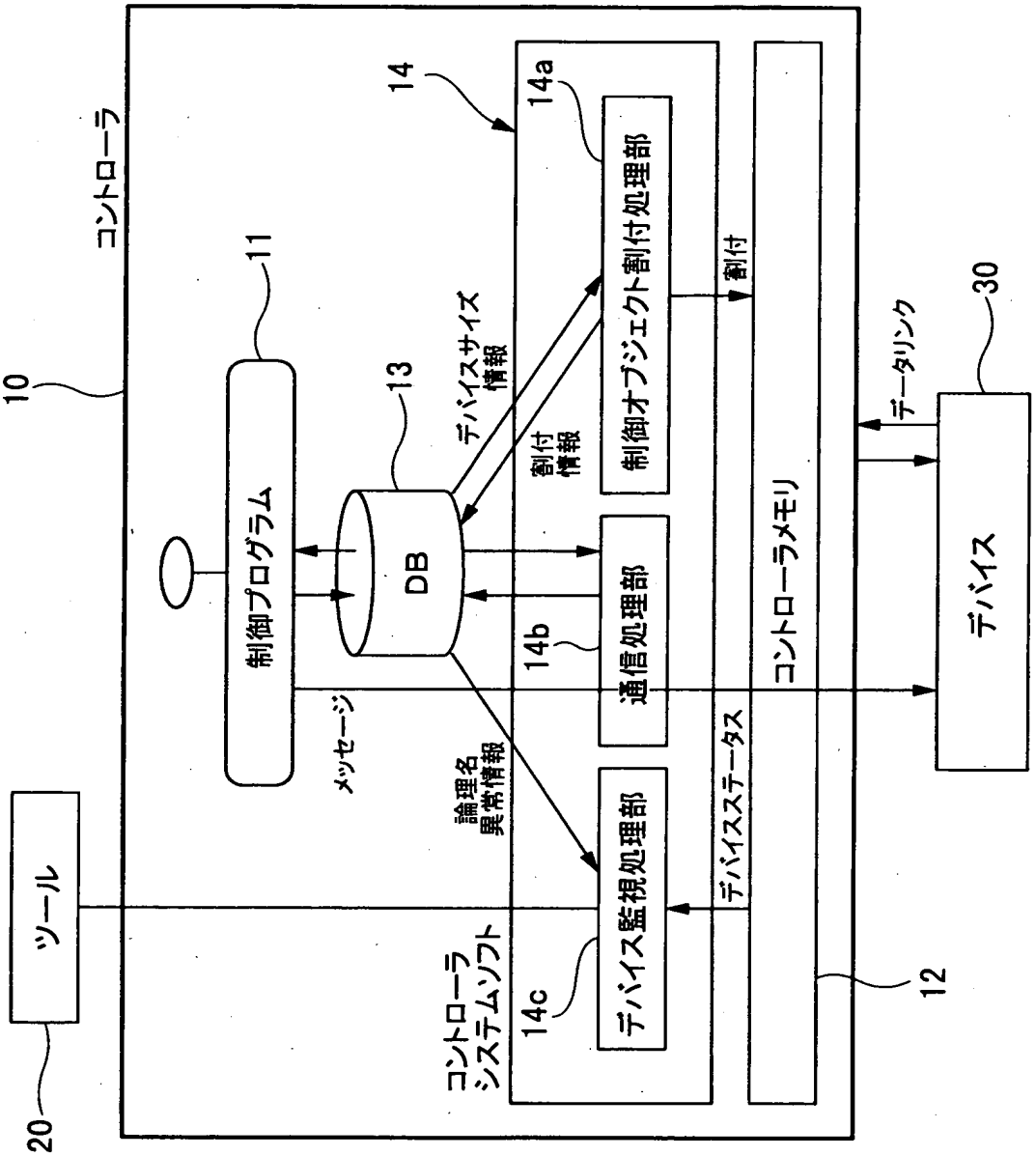
【図１】



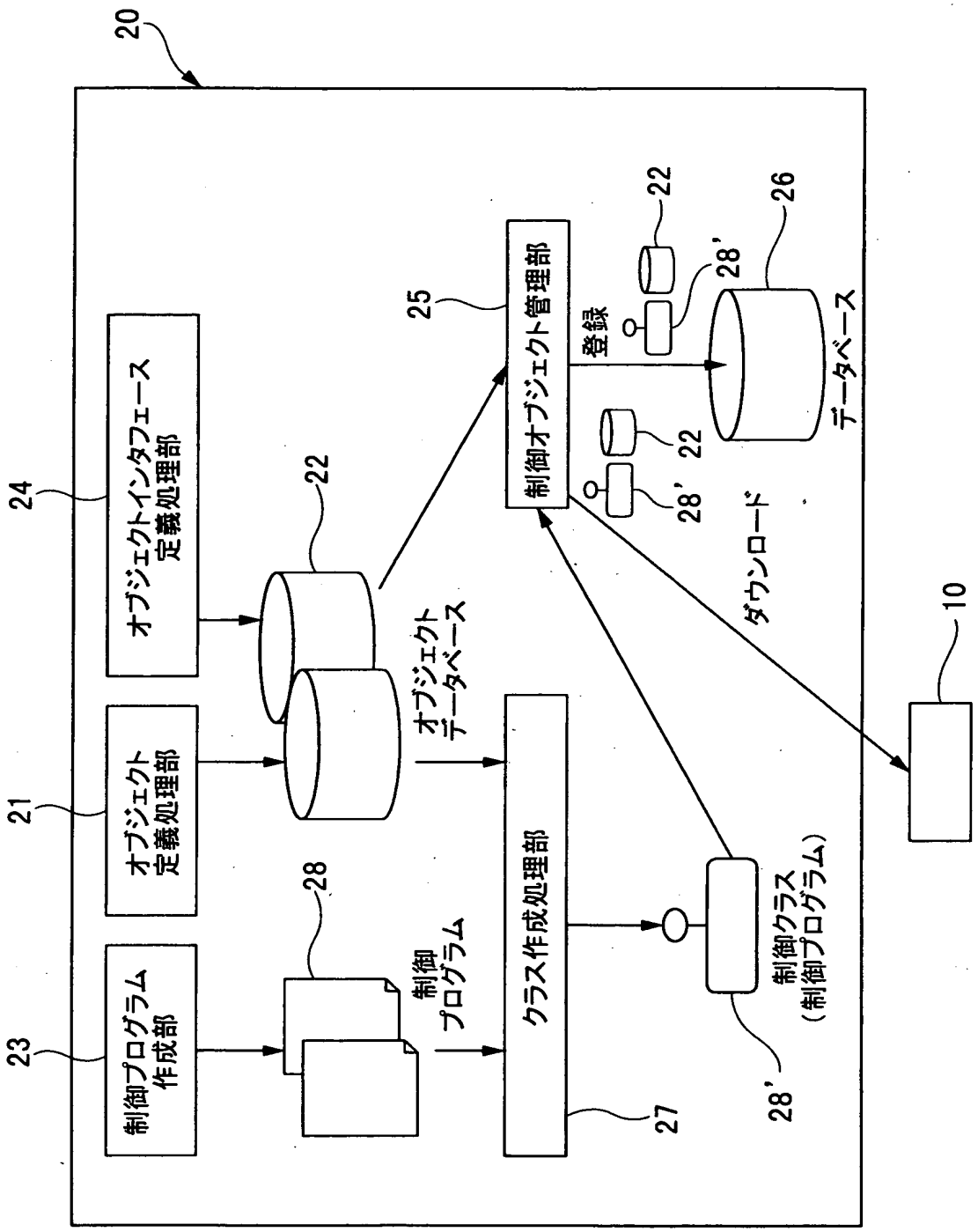
【図２】


9
4c
表示器
画面表示処理
7
5
通信処理
8c
8b
4
コントローラ
制御プログラム
通信処理2
デバイス
3
1
5
通信処理1
8a

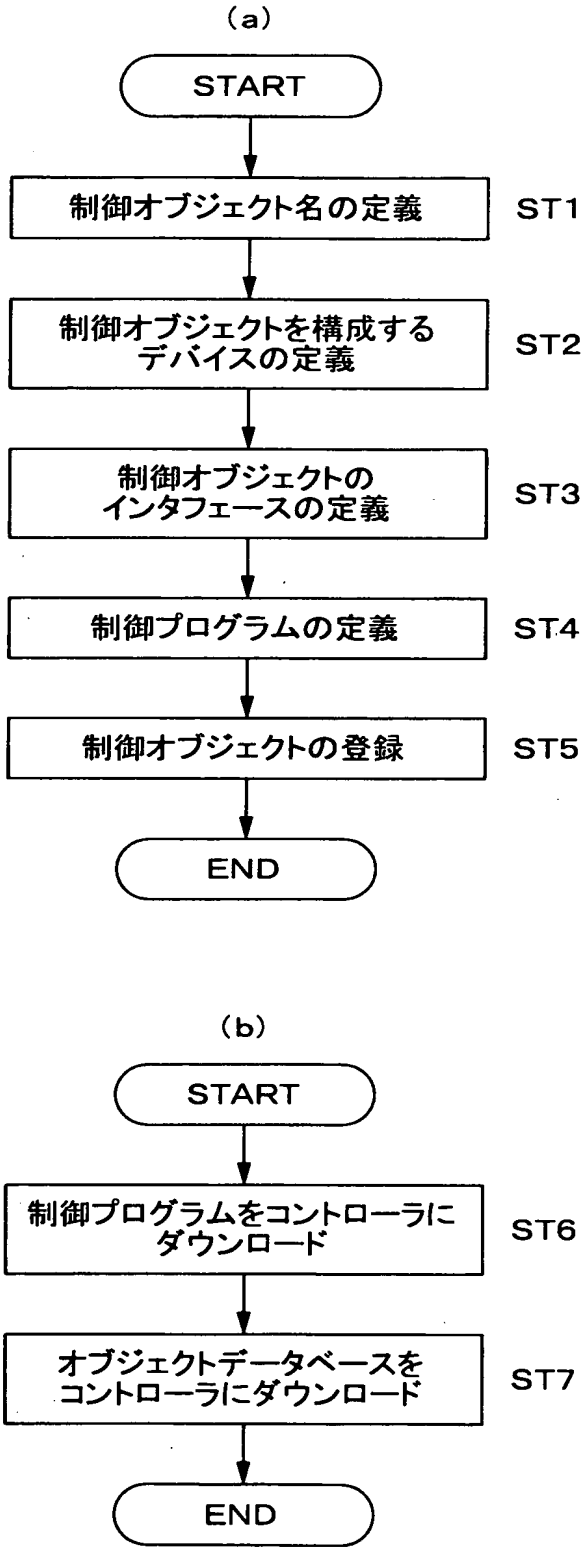
【図３】


コントローラ
10
11
制御プログラム
13
DB
デバイスサイズ情報
割付情報
14
14a
制御オブジェクト割付処理部
割付
14b
通信処理部
メッセージ
論理名異常情報
14c
デバイス監視処理部
デバイスステータス
コントローラシステムソフト
コントローラメモリ
12
データリンク
30
デバイス
ツール
20

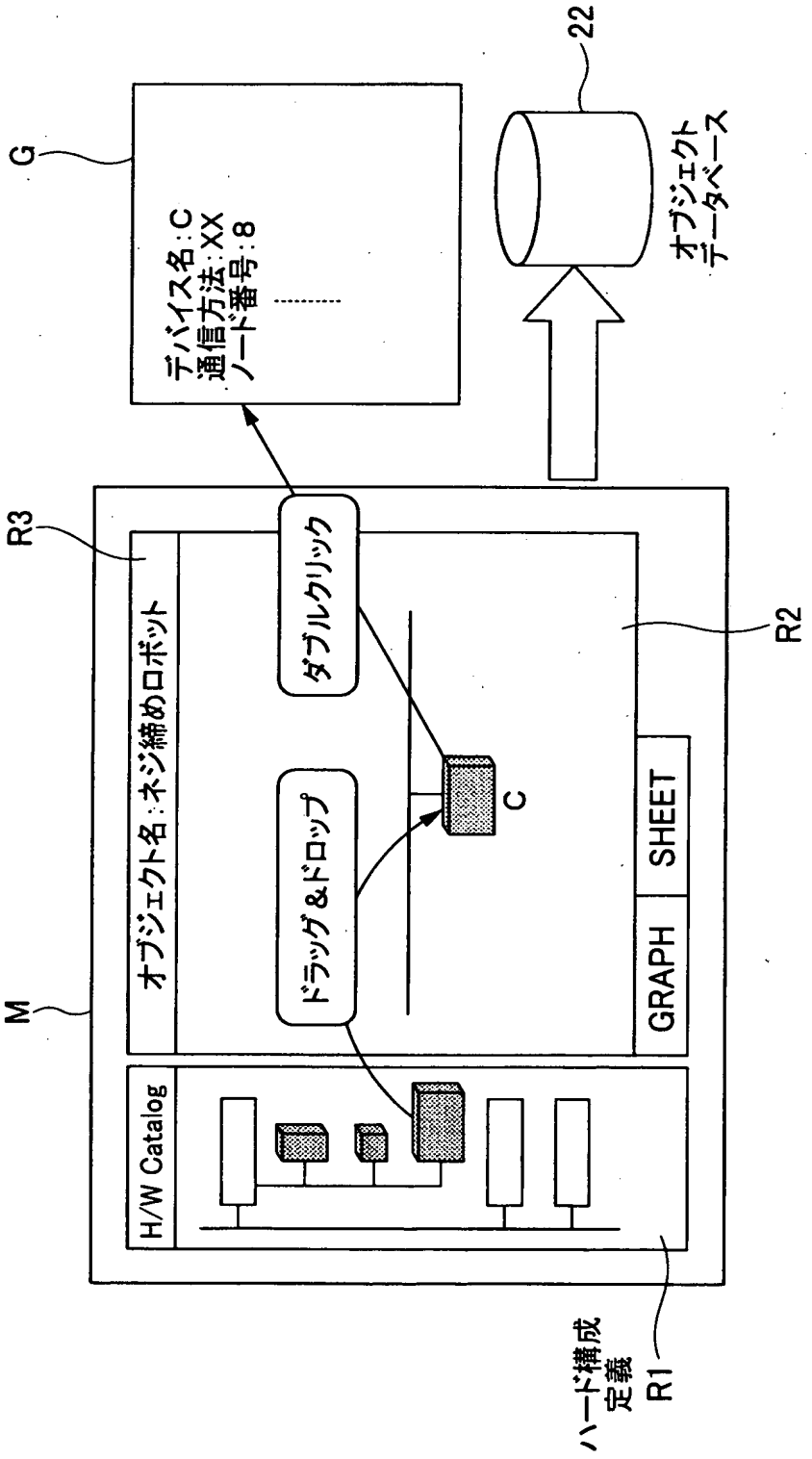
【図４】


20
オブジェクトインタフェース定義処理部
24
オブジェクト定義処理部
21
制御プログラム作成部
23
22
オブジェクトデータベース
28
制御プログラム
クラス作成処理部
27
25
制御オブジェクト管理部
登録
22
28'
26
データベース
22
28'
ダウンロード
10
制御クラス（制御プログラム）
28'

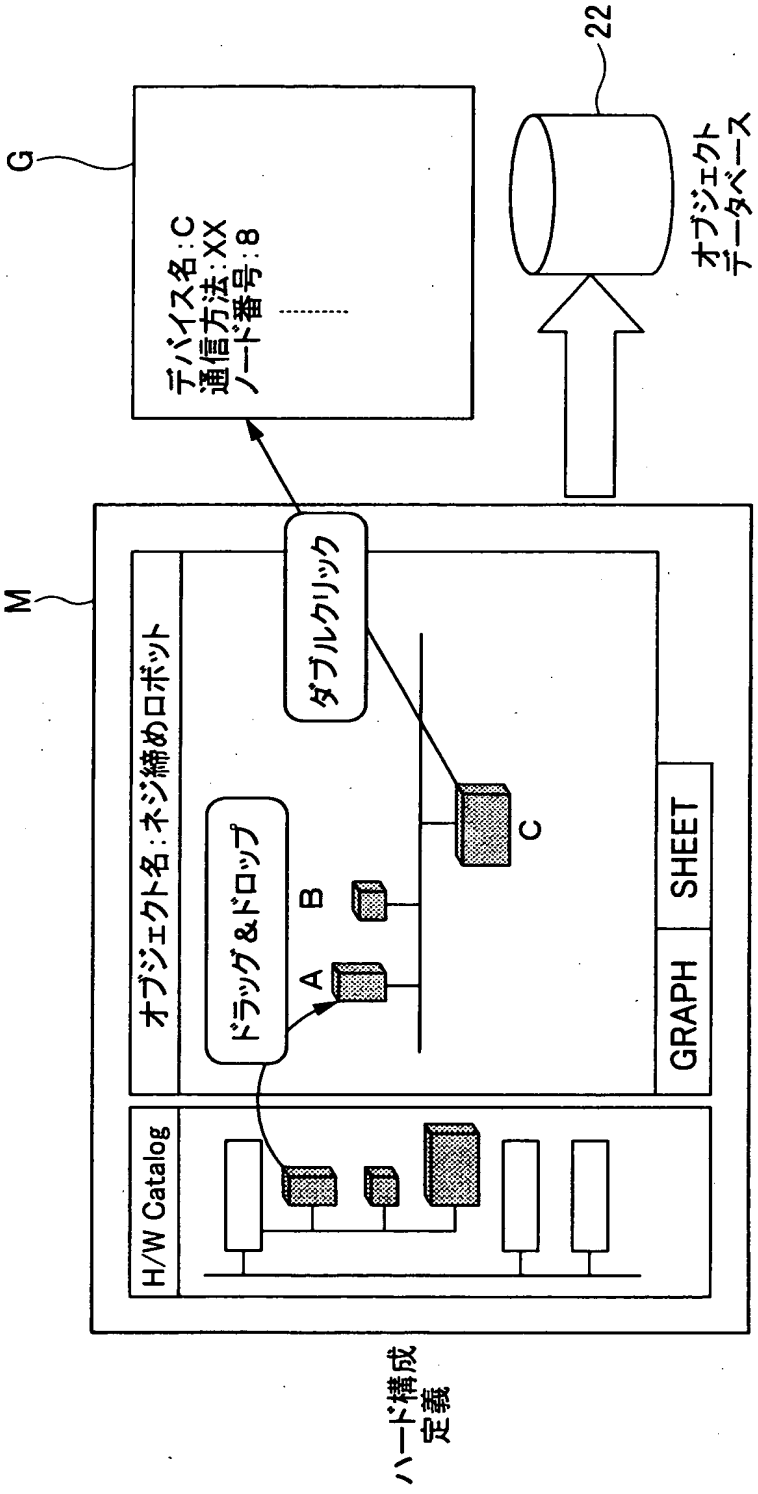
【図５】


(a)
START
制御オブジェクト名の定義
ST1
制御オブジェクトを構成する
デバイスの定義
ST2
制御オブジェクトの
インタフェースの定義
ST3
制御プログラムの定義
ST4
制御オブジェクトの登録
ST5
END
(b)
START
制御プログラムをコントローラに
ダウンロード
ST6
オブジェクトデータベースを
コントローラにダウンロード
ST7
END

【図６】


G
デバイス名:C
通信方法:XX
ノード番号:8
22
オブジェクト
データベース
R3
オブジェクト名:ネジ締めロボット
ダブルクリック
ドラッグ&ドロップ
C
R2
SHEET
GRAPH
M
H/W Catalog
ハード構成
定義
R1

【図７】


22
オブジェクト
データベース
G
デバイス名:C
通信方法:XX
ノード番号:8
.....
M
オブジェクト名:ネジ締めロボット
ダブルクリック
ドラッグ&ドロップ
A
B
C
H/W Catalog
GRAPH
SHEET
ハード構成
定義

【図８】

```
[Profile]
ObjName＝ネジ締めロボット

DeviceNum=1
  DevName0=C
  SerialNo0=
  NodeNo0＝8  //通信アドレス
  INSize0=2  //byte
  INadr0=    //コントローラメモリへの割付アドレス
  OUTSize0=2  //byte
  OUTadr0=  //コントローラメモリへの割付アドレス
  Communication0=0  //通信方法
```

【図９】

```
ObjName＝ネジ締めロボット

DeviceNum=3
  DevName0=C
  SerialNo0=
  NodeNo0＝8
  INSize0=2
  INadr0=
  OUTSize0=2
  OUTadr0=
  Communication0=0

  DevName1=A
  SerialNo1＝
  NodeNo1=3
  INSize1=4
  INadr1=
  OUTSize1=4
  OUTadr1=
  Communication1=0

  DevName2=B
  SerialNo2=
  NodeNo2＝1
  INSize2=1
  INadr2=
  OUTSize2=1
  OUTadr2=
  Communication2=0
```

【図１０】

```
[Profile]
ObjName＝ネジ締めロボット

DeviceNum=1
  DevName0=C
  SerialNo0=
  NodeNo0＝8  //通信アドレス
  INSize0=2  //byte
  INadr0=    //コントローラメモリへの割付アドレス
  OUTSize0=2  //byte
  OUTadr0=  //コントローラメモリへの割付アドレス
  Communication0=0  //通信方法

[Attribute]
IN_Num=2
  ValName0=IN_Param1  //変数名
  ValSize0=1  //1byte    //変数サイズ
  Adr0=0/0  //制御オブジェクト内アドレス(0CH目の0ビット)
  ValName1=IN_Param2  //変数名
  ValSize1=1  //byte    //変数サイズ
  Adr0=0/8  //制御オブジェクト内アドレス:0CH目の8ビット

OUT_Num=2
  ValName0=OUT_Param1
  ValSize0=1  //1byte    //変数サイズ
  Adr0=0/0  //制御オブジェクト内アドレス(0CH目の0ビット)
  ValName1=IN_Param2  //変数名
  ValSize1=1  //byte    //変数サイズ
  Adr0=0/8  //制御オブジェクト内アドレス:0CH目の8ビット

[Service]
```

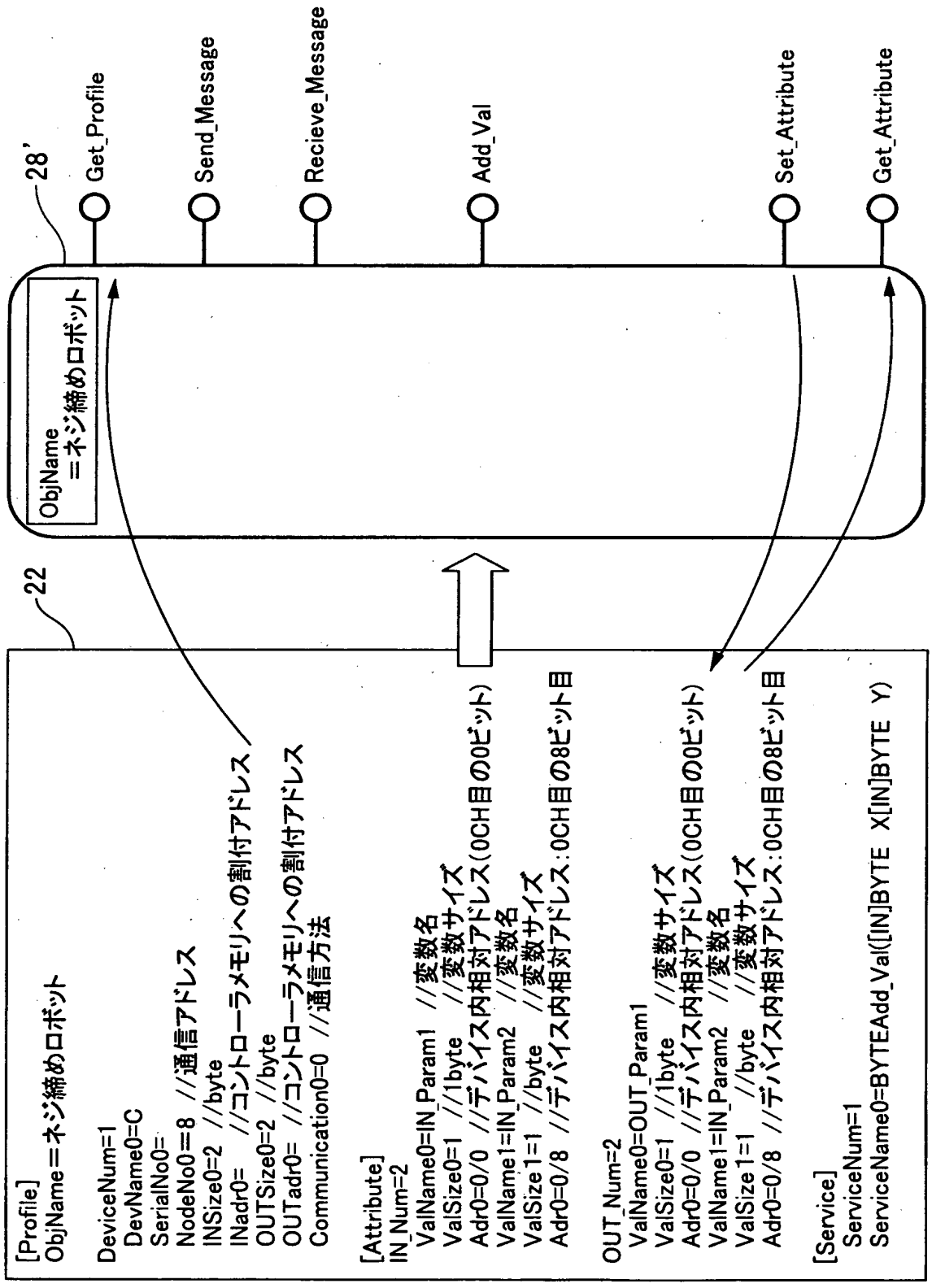
【図１１】

```
BYTE  Add_Val(BYTE  X, BYTE  Y)
{
  BYTE  A,B;C
  Get_Attribute("IN_param1", A);
  Get_Attribute("IN_param2", B);
  C=A+B;
  Set_Attribute("OUT_param1, C);
  Return  C
}
```

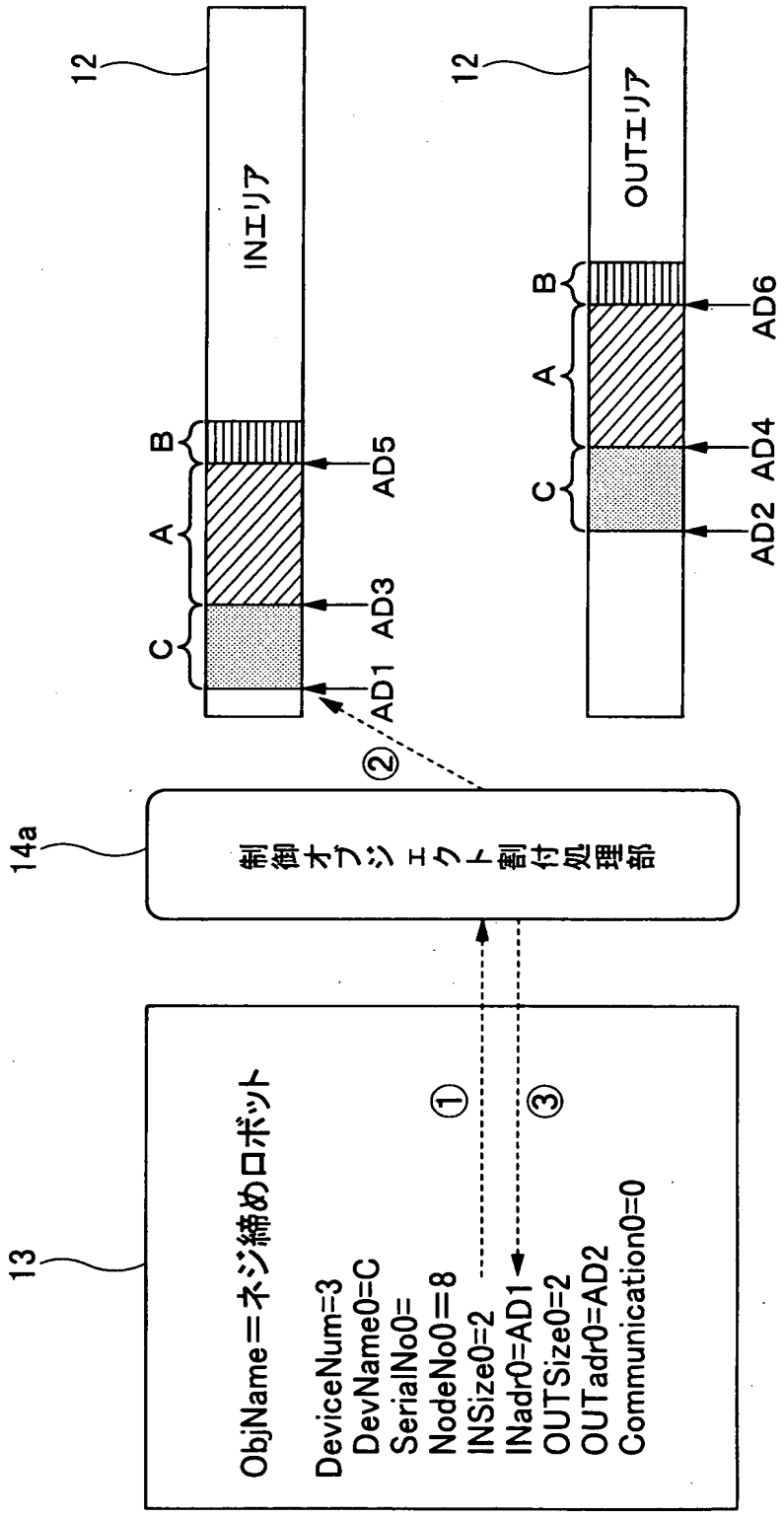
【図１２】

```
[Profile]
ObjName＝ネジ締めロボット

DeviceNum=1
  DevName0=C
  SerialNo0=
  NodeNo0＝8  //通信アドレス
  INSize0=2  //byte
  INadr0=    //コントローラメモリへの割付アドレス
  OUTSize0=2  //byte
  OUTadr0=  //コントローラメモリへの割付アドレス
  Communication0=0  //通信方法

[Attribute]
IN_Num=2
  ValName0=IN_Param1  //変数名
  ValSize0=1  //1byte    //変数サイズ
  Adr0=0/0  //デバイス内相対アドレス(0CH目の0ビット)
  ValName1=IN_Param2  //変数名
  ValSize1=1  //byte    //変数サイズ
  Adr0=0/8  //デバイス内相対アドレス:0CH目の8ビット目

OUT_Num=2
  ValName0=OUT_Param1
  ValSize0=1  //1byte    //変数サイズ
  Adr0=0/0  //デバイス内相対アドレス(0CH目の0ビット)
  ValName1=IN_Param2  //変数名
  ValSize1=1    //byte    //変数サイズ
  Adr0=0/8  //デバイス内相対アドレス:0CH目の8ビット目

[Service]
  ServiceNum=1
  ServiceName0=BYTEAdd_Val([IN]BYTE  X[IN]BYTE  Y)
```

【図１３】


28'
Get_Profile
Send_Message
Recieve_Message
Add_Val
Set_Attribute
Get_Attribute
ObjName
＝ネジ締めロボット
22
[Profile]
ObjName＝ネジ締めロボット
DeviceNum=1
DevName0=C
SerialNo0=
NodeNo0=8 //通信アドレス
INSize0=2 //byte
INadr0= //コントローラメモリへの割付アドレス
OUTSize0=2 //byte
OUTadr0= //コントローラメモリへの割付アドレス
Communication0=0 //通信方法
[Attribute]
IN_Num=2
ValName0=IN_Param1 //変数名
ValSize0=1 //1byte //変数サイズ
Adr0=0/0 //デバイス内相対アドレス(0CH目の0ビット)
ValName1=IN_Param2 //変数名
ValSize1=1 //byte //変数サイズ
Adr0=0/8 //デバイス内相対アドレス:0CH目の8ビット目
OUT_Num=2
ValName0=OUT_Param1
ValSize0=1 //1byte //変数サイズ
Adr0=0/0 //デバイス内相対アドレス(0CH目の0ビット)
ValName1=IN_Param2 //変数名
ValSize1=1 //byte //変数サイズ
Adr0=0/8 //デバイス内相対アドレス:0CH目の8ビット目
[Service]
ServiceNum=1
ServiceName0=BYTEAdd_Val([IN]BYTE X[IN]BYTE Y)

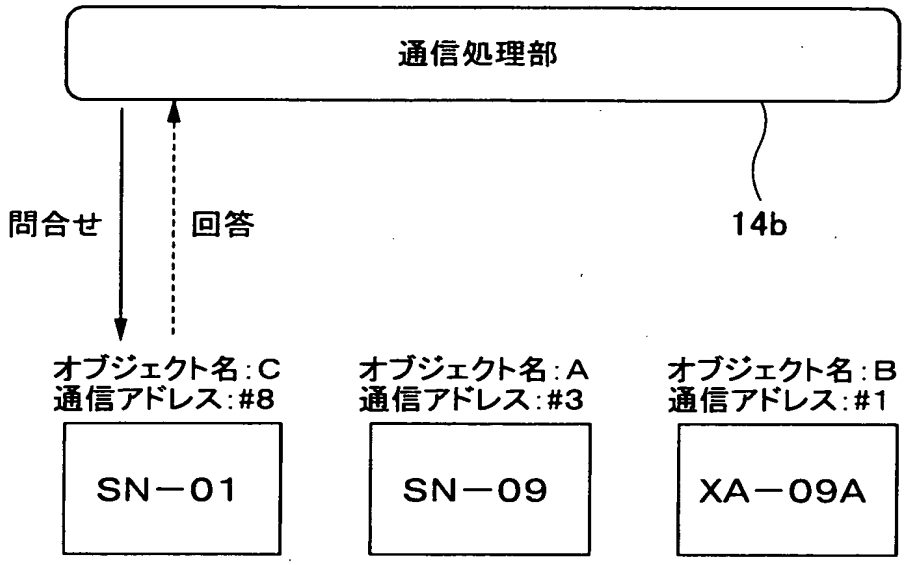
【図１４】


12
INエリア
B
A
C
AD5
AD3
AD1
12
OUTエリア
B
A
C
AD6
AD4
AD2
②
14a
13
ObjName＝ネジ締めロボット
DeviceNum=3
DevName0=C
SerialNo0=
NodeNo0＝8
INSize0=2
INadr0=AD1
OUTSize0=2
OUTadr0=AD2
Communication0=0
①
③

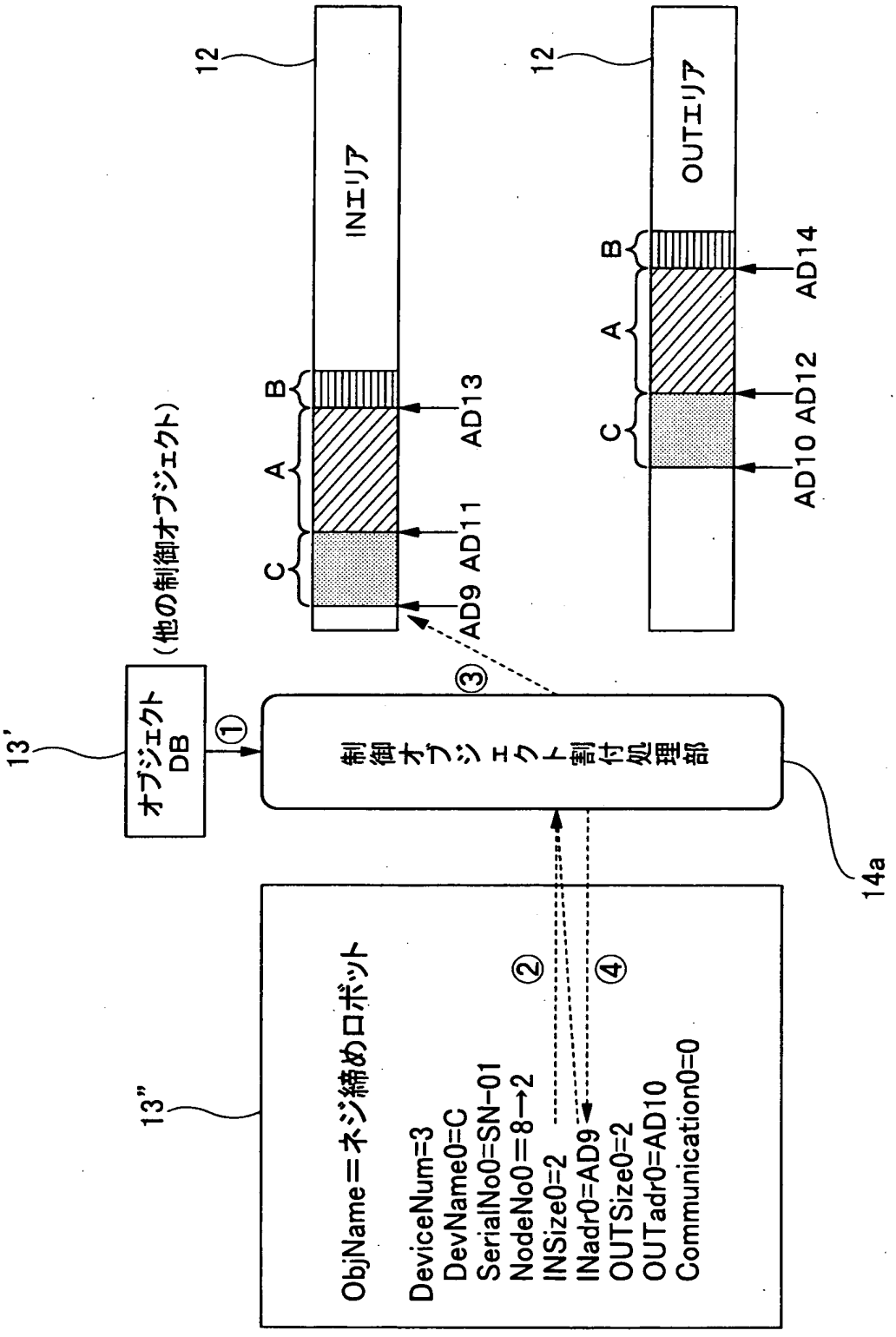
【図１５】

```
ObjName＝ネジ締めロボット

DeviceNum=3
  DevName0=C
  SerialNo0=SN-01
  NodeNo0＝8
  INSize0=2
  INadr0=AD1
  OUTSize0=2
  OUTadr0=AD2
  Communication0=0

  DevName1=A
  SerialNo1=
  NodeNo1＝3
  INSize1=4
  INadr1=AD3
  OUTSize1=4
  OUTadr1=AD4
  Communication1=0

  DevName2=B
  SerialNo2=
  NodeNo2＝1
  INSize2=1
  INadr2=AD5
  OUTSize2=1
  OUTadr2=AD6
  Communication2=0
```

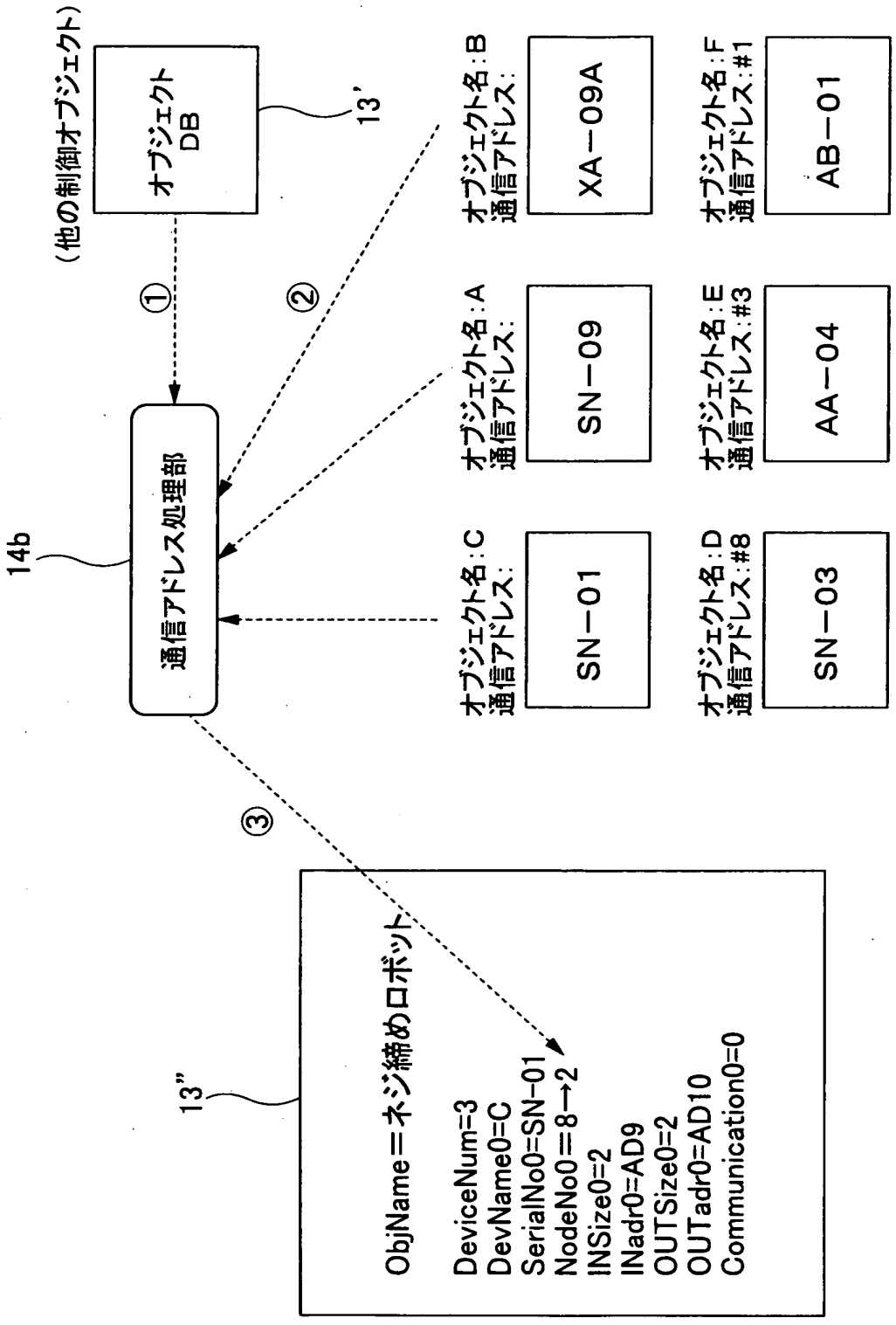
【図１６】


通信処理部
問合せ
回答
14b
オブジェクト名：C
通信アドレス：#8
SN－01
オブジェクト名：A
通信アドレス：#3
SN－09
オブジェクト名：B
通信アドレス：#1
XA－09A

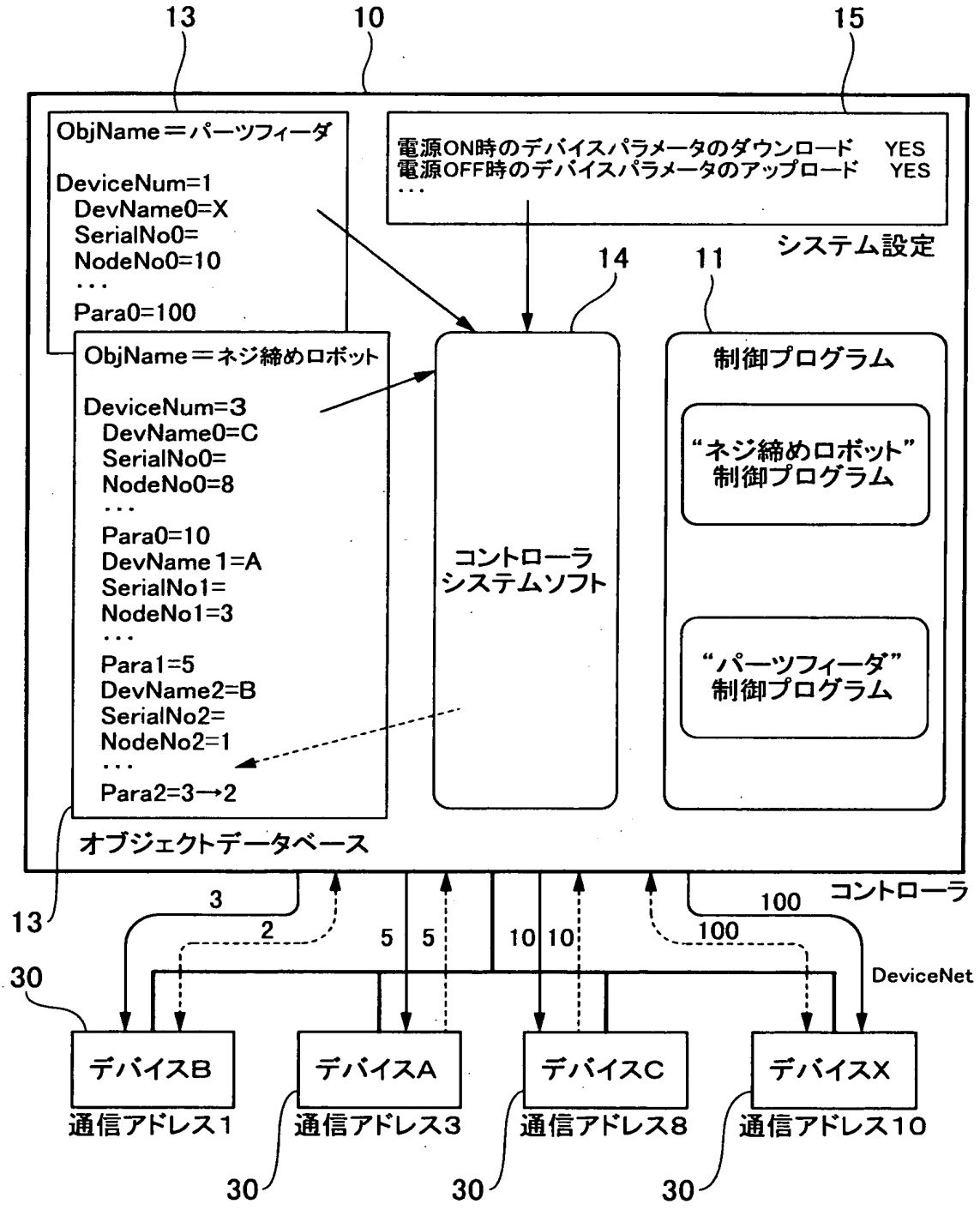
【図１７】


12
INエリア
B
A
C
AD13
AD11
AD9
12
OUTエリア
B
A
C
AD14
AD12
AD10
13'
オブジェクトDB
(他の制御オブジェクト)
①
③
制御オブジェクト割付処理部
14a
13"
ObjName=ネジ締めロボット
DeviceNum=3
DevName0=C
SerialNo0=SN-01
NodeNo0=8→2
INSize0=2
INadr0=AD9
OUTSize0=2
OUTadr0=AD10
Communication0=0
②
④

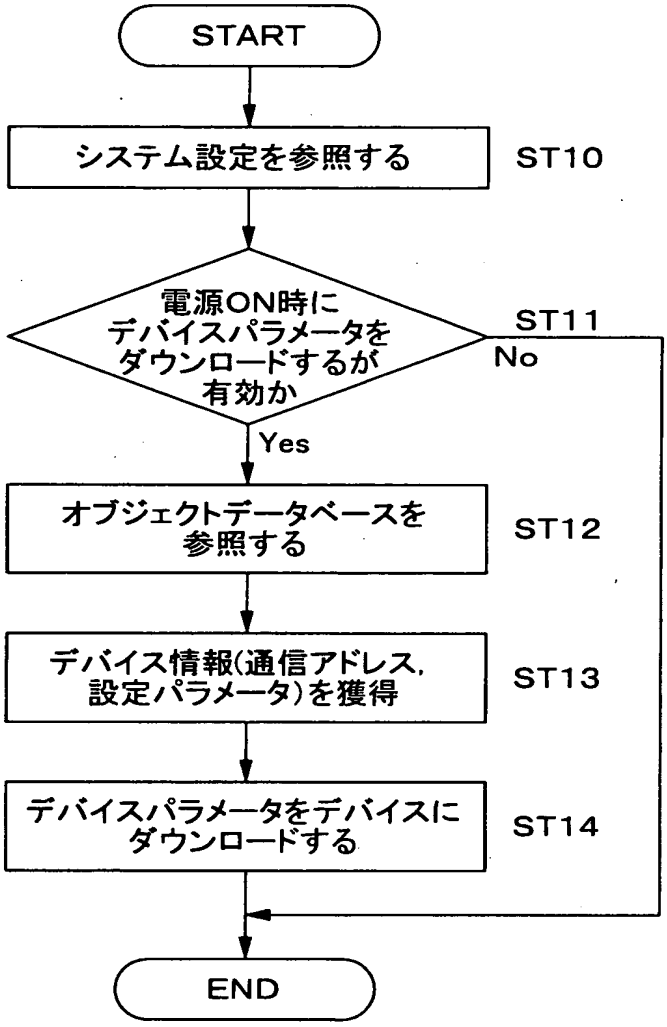
【図１８】


（他の制御オブジェクト）
オブジェクト DB
13'
①
②
③
14b
通信アドレス処理部
オブジェクト名:A 通信アドレス: SN−09
オブジェクト名:B 通信アドレス: XA−09A
オブジェクト名:C 通信アドレス: SN−01
オブジェクト名:D 通信アドレス:#8 SN−03
オブジェクト名:E 通信アドレス:#3 AA−04
オブジェクト名:F 通信アドレス:#1 AB−01
13"
ObjName＝ネジ締めロボット
DeviceNum=3
DevName0=C
SerialNo0=SN-01
NodeNo0=8→2
INSize0=2
INadr0=AD9
OUTSize0=2
OUTadr0=AD10
Communication0=0

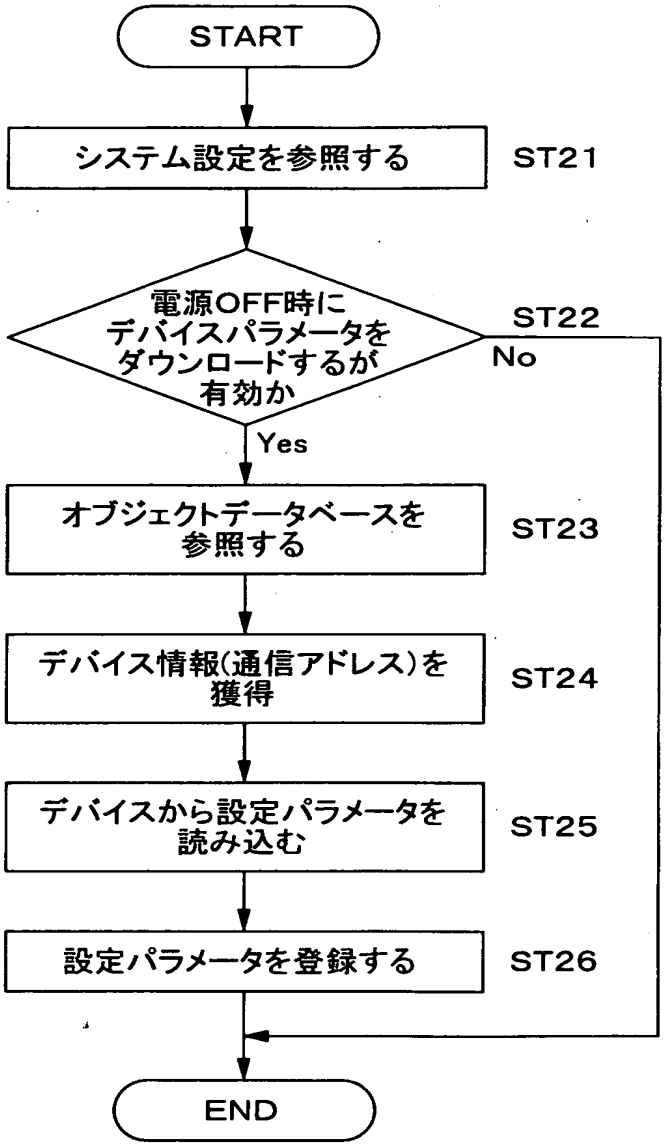
【図１９】


13
10
15
ObjName＝パーツフィーダ
DeviceNum=1
DevName0=X
SerialNo0=
NodeNo0=10
・・・
Para0=100
電源ON時のデバイスパラメータのダウンロード YES
電源OFF時のデバイスパラメータのアップロード YES
・・・
システム設定
14
11
ObjName＝ネジ締めロボット
DeviceNum=3
DevName0=C
SerialNo0=
NodeNo0=8
・・・
Para0=10
DevName1=A
SerialNo1=
NodeNo1=3
・・・
Para1=5
DevName2=B
SerialNo2=
NodeNo2=1
・・・
Para2=3→2
コントローラ
システムソフト
制御プログラム
“ネジ締めロボット”
制御プログラム
“パーツフィーダ”
制御プログラム
オブジェクトデータベース
コントローラ
13
3
2
5
5
10
10
100
100
DeviceNet
30
デバイスB
通信アドレス1
デバイスA
通信アドレス3
デバイスC
通信アドレス8
デバイスX
通信アドレス10
30
30
30

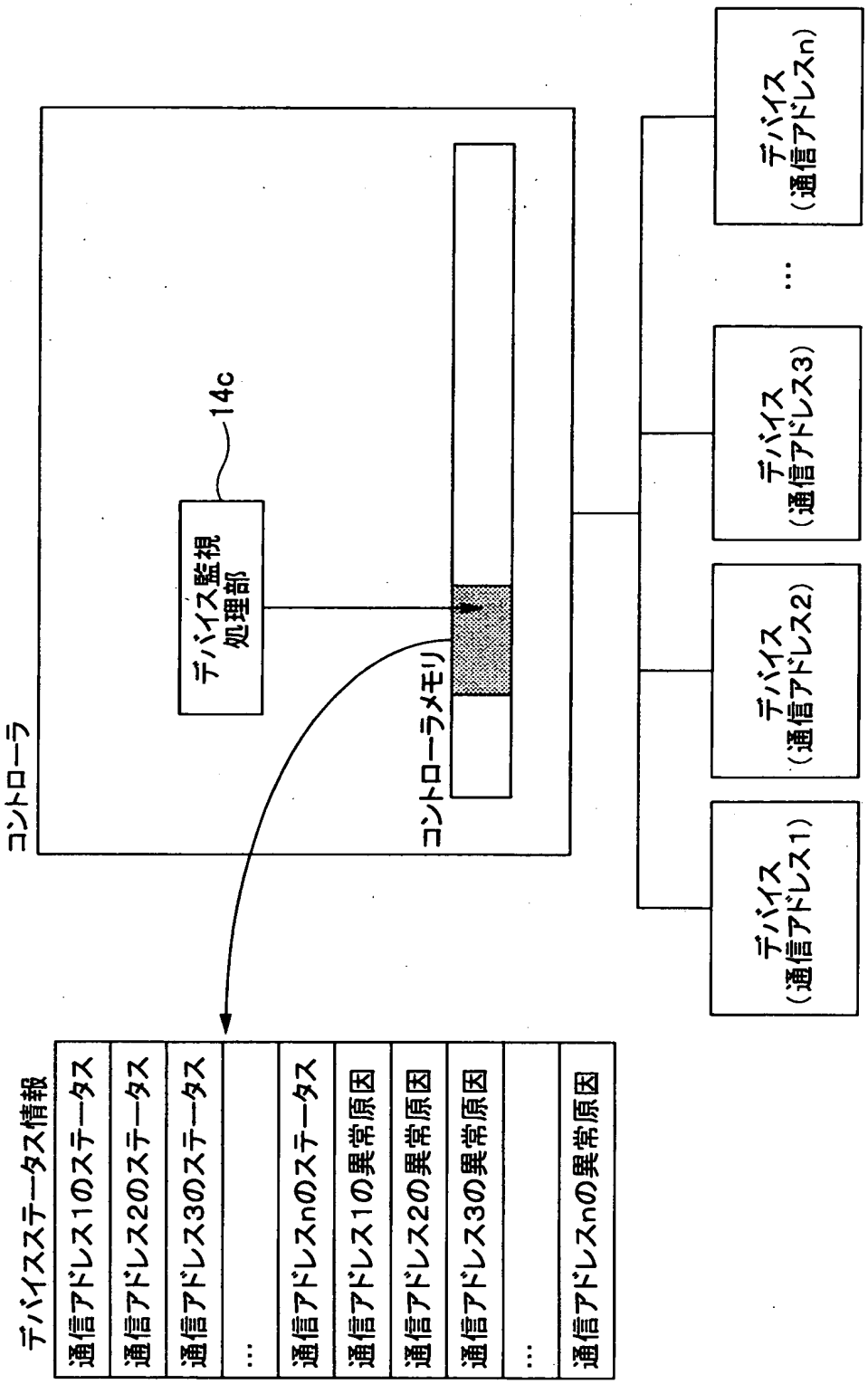
【図２０】


START
システム設定を参照する
ST10
電源ON時に
デバイスパラメータを
ダウンロードするが
有効か
ST11
No
Yes
オブジェクトデータベースを
参照する
ST12
デバイス情報(通信アドレス,
設定パラメータ)を獲得
ST13
デバイスパラメータをデバイスに
ダウンロードする
ST14
END

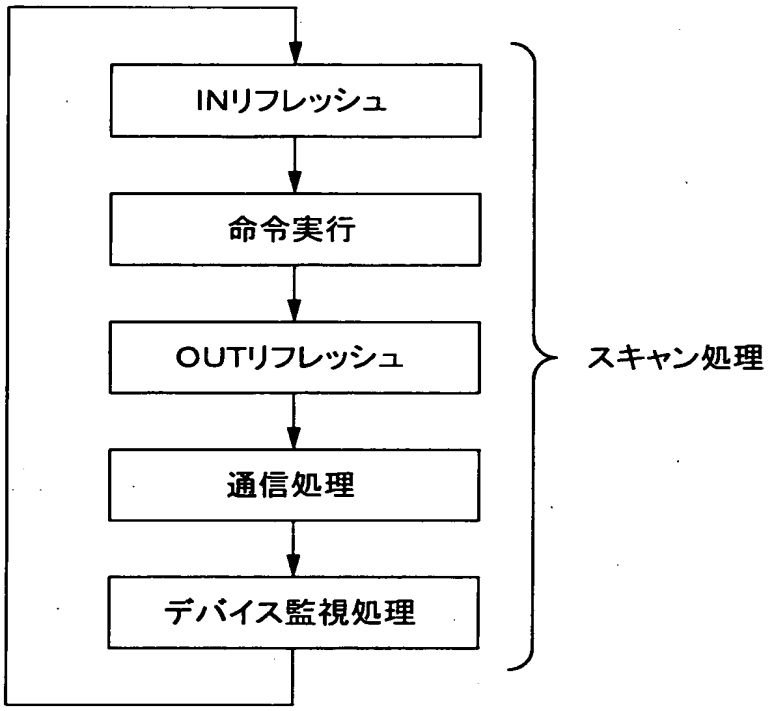
【図２１】


START
システム設定を参照する
ST21
電源OFF時に
デバイスパラメータを
ダウンロードするが
有効か
ST22
No
Yes
オブジェクトデータベースを
参照する
ST23
デバイス情報(通信アドレス)を
獲得
ST24
デバイスから設定パラメータを
読み込む
ST25
設定パラメータを登録する
ST26
END

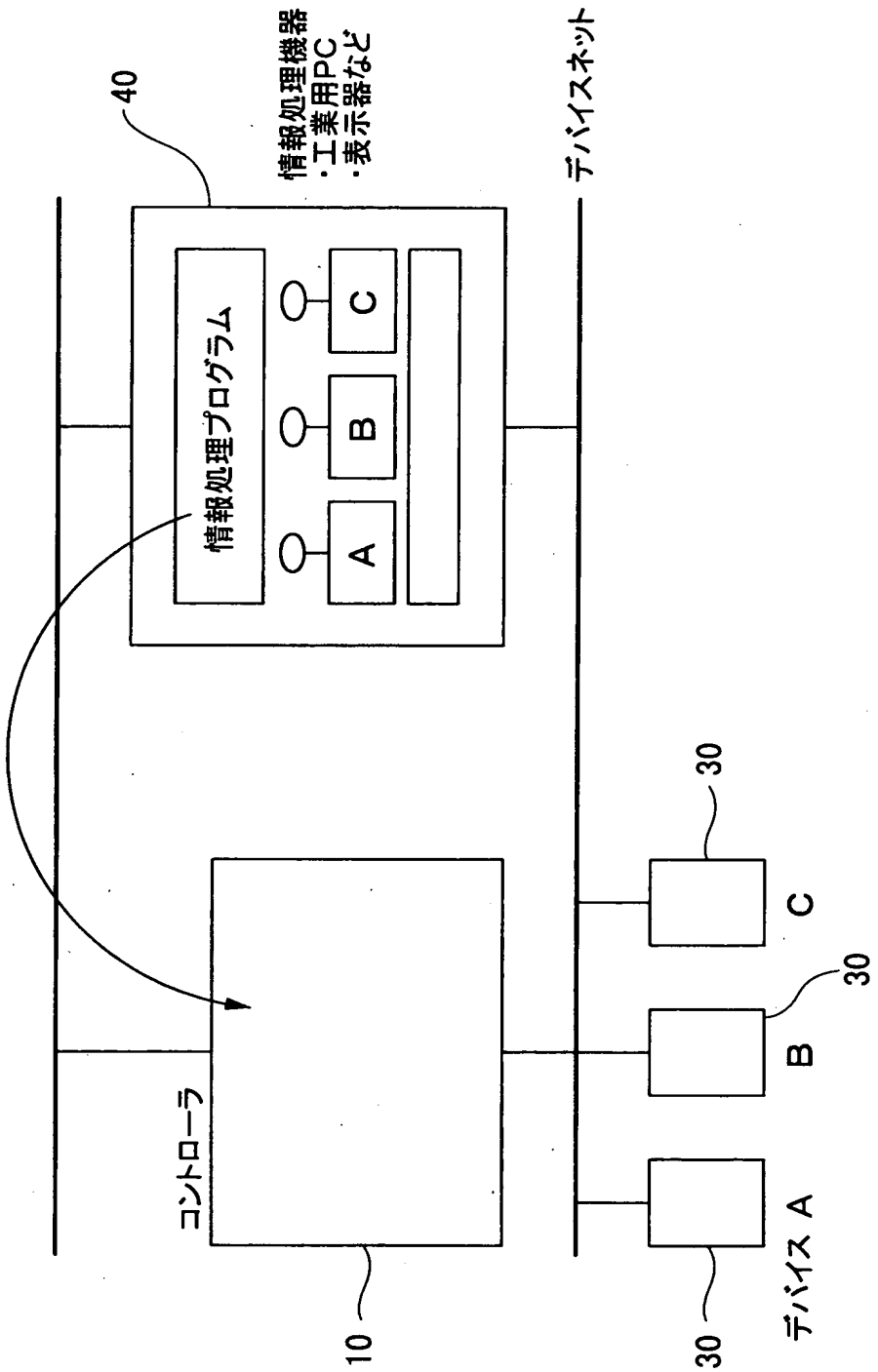
【図２２】


コントローラ
デバイス監視
処理部
14c
コントローラメモリ
デバイス
（通信アドレス1）
デバイス
（通信アドレス2）
デバイス
（通信アドレス3）
…
デバイス
（通信アドレスn）
デバイスステータス情報
通信アドレス1のステータス
通信アドレス2のステータス
通信アドレス3のステータス
…
通信アドレスnのステータス
通信アドレス1の異常原因
通信アドレス2の異常原因
通信アドレス3の異常原因
…
通信アドレスnの異常原因

【図２３】



【図２４】


40
情報処理機器
・工業用PC
・表示器など
デバイスネット
情報処理プログラム
C
B
A
コントローラ
10
30
C
30
B
30
デバイス A

【書類名】　　要約書

【要約】

【課題】　制御プログラムとデバイスを一体でオブジェクトとして扱うことができ、再利用が容易かつ確実に行えるコントローラを提供すること

【解決手段】　コントローラ１０とツール２０により構成され、コントローラには、デバイス３０が接続される。デバイスは制御プログラム１１により制御される。制御プログラムは、デバイス名で定義をし、デバイスの通信アドレス，割り付けられたメモリアドレス等とデバイス名を関連付けた関連情報をオブジェクトデータベース１３に格納する。デバイスにアクセスする際には、デバイス名をキーに関連情報を参照し、具体的なアクセス先を認識することによりデータの送受を行う。再利用する際には、制御プログラムの変更をすることなく、各アドレスを再利用先のものに修正するだけで対応できる。

【選択図】　　図３

出　願　人　履　歴　情　報

識別番号　　〔０００００２９４５〕

１．変更年月日　　２０００年　８月１１日
［変更理由］　住所変更
住　所　京都市下京区塩小路通堀川東入南不動堂町８０１番地
氏　名　オムロン株式会社